# APPROCHE DIRECTE DE L'ESTIMATION AUTOMATIQUE DE L'ORIENTATION 3D D'IMAGES

Mahzad Kalantari

UNIVERSITÉ DE NANTES

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIE
DE L'INFORMATION ET DES MATHÉMATIQUES


Année 2009

## Thèse de doctorat de l'université de Nantes

Spécialité : AUTOMATIQUE ET INFORMATIQUE APPLIQUÉE

*Présentée et soutenue publiquement par*


## Mahzad KALANTARI


*le 4 septembre 2009*
*à l'École Nationale des Sciences Géographiques*

# APPROCHE DIRECTE DE L'ESTIMATION AUTOMATIQUE DE L'ORIENTATION 3D D'IMAGES

## jury

| | | |
|---|---|---|
| Président : | M. KASSER | *Professeur des universités, École Nationale des Sciences Géographiques, IGN* |
| Rapporteurs : | P. GRUSSENMEYER | *Professeur des universités, INSA Strasbourg, UMR MAP 694* |
| | J. LOUCHET | *HDR, chercheur INRIA, ARTENIA* |
| Examinateurs : | L. POLIDORI | *Professeur titulaire de chaire, École Supérieure des Géomètres Topographes, CNAM* |
| | F. JUNG | *Directeur adjoint DDE 76* |
| | JP. GUÉDON | *Professeur des universités, École polytechnique de l'université de Nantes* |
| Invité : | Y. EGELS | *Ingénieur général honoraire* |


**Directeur de thèse : JP. GUÉDON**
Laboratoire : Institut de Recherche en Communications et Cybernétique de Nantes
**Co-encadrant : F. Jung**

Composante de rattachement du directeur de thèse : École polytechnique de l'université de Nantes



$N^\circ$ ED : 503-061

***Ecrire liberté sur le bord d'une plage, c'est déjà avoir la liberté de l'écrire. Même si la mer efface ce mot : la liberté demeure.***

**Jean-Michel Wyl**

تیغ دادن در کفه زنگي مست  به که آید علم ناکس را به دست

مولانا

دي شیخ با چراغ همي گشت گرد شهر

کز دیو و دد ملولم و انسانم آرزوست

گفتند یافت مي نشود جسته ایم ما

گفت آن که یافت مي نشود آنم آرزوست

مولانا

*ΑΓΕΩΜΕΤΡΗΤΟΣ ΜΗΔΕΙΣ ΕΙΣΙΤΩ*

***Que nul n'entre s'il n'est géomètre***

# Remerciements

Je l'ai, comme beaucoup d'autres étudiants, découvert en cours de route : une thèse, d'après ce que j'ai pu voir, n'est que rarement un long fleuve tranquille. Celle-ci en tous cas ne l'a pas été, elle m'a amené à bouger beaucoup, et à changer trois fois de laboratoire en guère plus de trois années. J'ai donc un nombre considérable de personnes à remercier de m'avoir aidée, à un titre ou à un autre. Parfois même sans qu'ils s'en rendent compte, elles ont toutes contribué au succès de ce travail.

– Bien évidemment mon directeur de thèse, JeanPierre Guédon, sans qui tout ceci se serait certainement mal terminé et qui a supporté stoïquement mes réactions, certes toutes justifiées, mais quand même... et puis mon co-directeur Franck Jung, qui as toujours été à mon écoute, et avec qui j'ai eu beaucoup de plaisir à travailler.

– et puis, tout particulièrement, les deux rapporteurs de ma thèse, Jean Louchet et Pierre Grussenmeyer, qui ont accepté de consacrer une partie de leurs congés à analyser de près mes travaux.

– mes professeurs et collègues en Iran, et parmi eux le professeur Ali Azizi qui m'a donné le goût de la photogrammétrie, sans ses cours je n'aurai jamais poursuivi dans cette voie, et Amir Hashemi qui m'a guidé dans les complexités des bases de Gröbner. Et puis en France, Marc Sigelle avec qui les discussions ont toujours très enrichissantes.

– à l'IVC, qui m'a toujours reçu dans d'excellentes conditions et qui, comme seuls les vrais laboratoires s'obligent à le faire, a financé mes conférences à l'étranger ; Merci à Patrick, et toute l'équipe du labo, Nicolas, Guillaume, Benoit, Sylvain, Florent, Eloise, Stéphane, ... avec une pensée toute particulière pour Aurore et son dynamisme.

– A l'ESGT au cours de ma première année, Elisabeth Simonetto et Eric Labergerie pour nos travaux en commun en photogrammétrie, Rani El Méouche qui a toujours été à mes cotés et avec qui j'ai partagé les galères de la vie de thésard, Christophe Proudhom pour avoir toujours tenté de concilier des impératifs de planification avec mes demandes, et Laurent Polidori qui a réussi à honorer, même tardivement, les promesses du CNAM, et qui surtout a accepté de siéger à mon jury de thèse et d'y apporter sa grande expertise.

– Au MATIS ensuite, où j'ai rencontré des gens formidables, au départ simples collègues, des relations très amicales se sont vite installées. Je pense particulièrement à Erwan, qui a toujours été présent quand j'avais besoin de lui. Et puis Nesrine, et sa bonne humeur. Je pense aussi à Fadi et son humilité malgré ses impressionnants scores de chercheur, ainsi que sa gentillesse.

... et bien sur, d'autres membres du laboratoire : Arnaud, Bertrand, Jean-Pierre, Karim, Jean-Pascal ainsi que Jean-Michael des IS, qui n'ont cessé d'apporter leur soutien actif à mes recherches, et puis aussi Mounia, Marie-Claude, François et Alain qui m'ont entouré de leurs conseils et ont œuvré pour aplanir les difficultés autant qu'ils l'ont pu.

– A l'ENSG enfin, qui m'a permis de façon quasi inespérée de terminer mes travaux dans d'excellentes conditions, Michel Kasser (qui a aussi accepté la charge de présider mon jury de

thèse), Véronique Lemaire qui m'a gentiment accueilli dans son département, et Olivier Dissard, Olivier de Joinville, Sylvie Ligé, et bien d'autres que je ne peux tous nommer, m'ont reçu dans un cadre chaleureux et parfaitement adapté. Une petite pensée aussi pour les membres du laboratoire LAREG, avec qui j'ai partagé d'innombrables pauses : Leila, Pierre, Alvaro et bien sur Christiane.

... dommage que l'accès aux abonnements électroniques, auquel j'ai pu avoir accès dans d'autres laboratoires, ne soit pas encore possible à l'IGN. Je remercie donc d'autant plus Anne Berry et le CDOC de l'IGN en général, qui a toujours été très réactif pour me fournir les articles papier dont j'avais besoin pendant mes recherches.

... une pensée particulière à Patricia Bordin, et nos pauses café-crème. Merci à son écoute et son soutien, et à ses conseils dans des moments les plus difficiles.

... et un petit clin d'øeil à Ismaila,

Je me dois aussi de remercier le Conseil Général de la Sarthe, la Communauté Urbaine du Mans, le Conseil Régional des Pays de Loire, puis le programme Terra Numerica et l'IGN, pour avoir soutenu financièrement mes recherches. J'associe à ces remerciements la SFPT qui m'a honoré d'un prix prestigieux pour fêter son 50ème anniversaire.

Et bien sur je remercie mes parents qui m'ont soutenue sur tous les plans, malgré la complexité du contexte international. Ils m'ont appris que la science est un investissement éternel. Et aussi mon frère Amir Shahriar, et mon ami Reza. C'est à eux que je dédie cette thèse.

# Résumé

**APPROCHE DIRECTE DE L'ESTIMATION AUTOMATIQUE DE L'ORIENTATION 3D D'IMAGES**

Récemment, la géomatique grand public s'est emparée de la représentation 3D des bâtiments. Le besoin d'acquérir des images et de les restituer en 3D, sous forme de maquettes parfaitement fidèles à la réalité, est ainsi devenu immense. On a donc vu depuis une décennie, se construire des véhicules capables de photographier en stéréoscopie des villes entières, et il a fallu concevoir les algorithmes capables de traiter ces énormes quantités d'images. Très naturellement, les industriels en charge de ces problèmes se sont tournés vers les outils de vision par ordinateur et de robotique, très bien adaptés aux calculs temps réel, oubliant l'essentiel de l'héritage de la photogrammétrie, orientée quant à elle vers une extrême précision, jugée ici comme une moindre priorité. Néanmoins, les algorithmes publiés en vision par ordinateur présentaient de réels défauts lorsqu'ils étaient appliqués à des surfaces planes alors que ce cas est extrêmement courant dans des scènes urbaines pour traiter les façades de bâtiments.

Les recherches que nous avons menées ont porté sur la recherche de solutions nouvelles, capables d'exploiter les spécificités de telles images : tout d'abord, nos travaux ont cherché à accélérer l'orientation relative des images, en tirant bénéfice des points de fuite figurant dans celles-ci. Pour ce faire, de nouvelles méthodes d'extraction automatique de ces points ont été mises au point et évaluées plus performantes que celles disponibles jusqu'ici.

Ensuite, nos recherches ont porté sur les moyens de corriger le défaut évoqué précédemment pour les surfaces planes, et de nouveaux algorithmes capables de donner en temps quasi-réel de bonnes solutions d'orientation relative pour de telles scènes ont été développés. A cette fin, de nouveaux outils mathématiques ont été utilisés : les bases de Gröbner. En rupture complète avec les solutions linéaires habituelles, ils permettent en effet une résolution directe des équations sous leur forme polynomiale. Ils évitent de passer par l'habituelle linéarisation, qui nécessitait une solution approchée parfois difficile à trouver dans les usages de photogrammétrie terrestre. Finalement, nos travaux ont porté sur les moyens d'accélérer les méthodes d'orientation relative en exploitant opportunément la connaissance de la direction verticale, obtenue par exemple à l'aide du nouvel algorithme de détection des points de fuite.

Au total, la thèse présente une remise à plat générale des solutions permettant l'orientation et la localisation de tout un ensemble d'images.

**Mots-clés** : Photogrammétrie, vision par ordinateur, détection de points de fuite, bases de Gröbner, résolution directe

# Abstract

**A DIRECT APPROACH TO AUTOMATIC ASSESSMENT OF 3D IMAGES ORIENTATION**

In the recent years, the geomatics used by large public has been seized by the 3D representation of buildings and even of whole cities. The need to acquire some images and to draw them in 3D, as models perfectly representative of the reality became thus very large. One saw therefore, since one decade, the setting up of vehicles able to get stereoscopic images of complete cities, and thus it became necessary to forge the algorithms processing these huge amounts of images. In a natural manner, those in charge of these problems turned toward the tools used in computer vision and robotics, fully adapted to real time, and thus forgetting most of the inheritance of photogrammetry, that is oriented towards an extreme precision (judged here as a low priority). Nevertheless, the available algorithms in computer vision presented some shortcomings when they were applied to plane surfaces which is extremely current in urban scenes to cope with the facades.

Our research have tried to find new solutions, able to exploit the specificities of such images : first of all, the work aimed at accelerating their relative orientation, taking benefit from the detected vanishing points. To achieve this goal new methods of automatic extraction of these points have been finalized, assessed as more efficient than those previously available. Then, researches have been led so as to correct the previous drawback for plane surfaces, and new algorithms able to provide in quasi-real time good solutions of relative orientation for such situations have been developed. To reach this goal, new mathematical tools permitting a complete rupture with traditional ones have been used : the Gröbner bases. They allow a direct resolution of the equations under their polynomial shape, without using a classical linearization, that required an approach solution sometimes difficult to find in terrestrial photogrammetry.

Finally, the work has been dealing with the means to accelerate the methods of relative orientation using the specific knowledge of the vertical direction, given for example with the help of the new algorithm of detection of the vanishing points. In short, this works aims to present a general re-analysis of the solutions allowing the orientation and localization of a whole set of images.

**Keywords** : Photogrammetry, computer vision, vanishing points detection, Gröbner basis, direct resolution

# Table des matières

# 1
# Introduction

## A la conquête de la troisième dimension

Il semble bien que dès qu'il y a eu des hommes sur Terre, ils ont été puissamment attirés par la maîtrise de l'image et, plus encore, de la représentation en 3D des objets et du monde environnant. De ces époques très reculées, les archéologues ont pu retrouver par exemple des gravures rupestres, tracés plus ou moins figuratifs. Ainsi de trop rares exemples, qui ont survécu à plusieurs dizaines de milliers d'années, nous viennent des grottes ornées. Et puis, parfois aussi dans de tels sites, ils nous ont fait connaître quelques productions d'artistes, déjà capables de réaliser des figurines en relief extrêmement abouties.

(a) (b) (c)

**Fig. 1.1.** (a) Grotte de Lascaux, 15 000 ans av. JC. (b) Félin anthropomorphe, de Hohlenstein-Stadel (Allemagne), vers 30 000 ans av. JC. (c) Bisons d'argile, grotte du Tuc d'Audoubert, vers 15 000 - 10 000 av. JC. Quelques exemples célèbres issus de la préhistoire, premières images 2D, premières représentations 3D.

L'étape suivante, c'est il y a cinq ou six millénaires, à l'aube de la civilisation, où l'on a commencé à produire des images 2D extrêmement soignées, dont la fonction était celle des appareils photo d'aujourd'hui : garder une archive pas trop onéreuse d'une personne, d'une scène, etc... Néanmoins, il s'agissait d'une œuvre très coûteuse, ne serait-ce que par l'extrême qualification nécessaire pour que le peintre obtienne des résultats satisfaisants. Lorsque des

moyens financiers encore plus élevés étaient disponibles, des images en 2,5 D[1], ont été parfois employées (figures en ronde-bosse, bas-reliefs).

(a) (b) (c) (d)

**Fig. 1.2.** (a) et (b) Persépolis-Shiraz (Iran, 500 av. JC.). (c) Ramses III (Egypte, 1200 av. JC.). (d) Une scène présentant un géographe sur le campanile du *Duomo* de Florence. A mi-chemin entre l'image classique et les statues, comparativement très coûteuses à fabriquer, voici des exemples anciens de représentations en 2,5 D, de lecture plus facile et de meilleure conservation future que des peintures.

Et bien entendu, lorsqu'il s'agissait d'enjeux importants, justifiant des dépenses considérables (art sacré, personnages de haut rang), les artistes pouvaient aussi produire du vrai 3D : terme moderne barbare pour parler de la sculpture, dont d'extraordinaires chefs d'œuvre ont

1. Entre les termes 2D (une image) et 3D (p. ex. une statue, une maquette, une présentation sur écran permettant de montrer de façon dynamique une scène avec une grande variété de points de vues, ou encore une présentation permettant la stéréoscopie), le terme de 2,5 D est employé pour désigner des représentations où spécifiquement pour un point donné du plan, une seule valeur d'altitude est représentée. En 2,5 D, on ne peut pas représenter l'intérieur d'un bâtiment, les toitures ne peuvent déborder sur les façades, etc... ce n'est donc pas de la vraie 3D, d'où ce terme conventionnel.

pu parvenir jusqu'à nous.

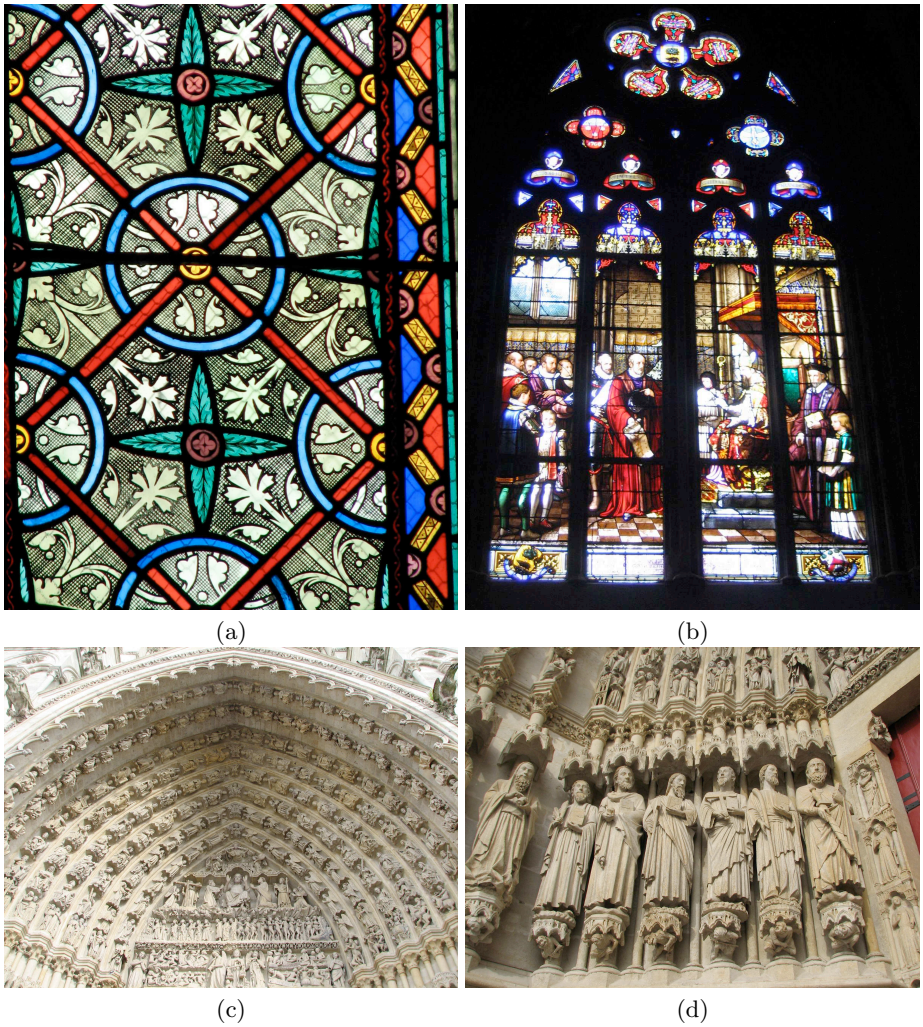
(a) (b)

**Fig. 1.3.** (a) La Victoire de Samothrace, fin IV$^{ème}$ début III$^{ème}$ av. JC, musée du Louvre. (b) Statue de Pinedjem I, temple de Karnak, Egypte. Les civilisations antiques qui ont fait travailler par leurs artistes des matériaux très résistants nous ont légué d'extraordinaires représentations 3D. Compte tenu de l'importance du travail nécessaire, il s'agit presque toujours de personnages humains de haut rang, ou divins, ou encore d'allégories.

Entre la 2D, comparativement "bon marché" et la 3D, il y a toujours eu cette hiérarchie des coûts. On la retrouve évidemment de mieux en mieux lorsqu'on se rapproche de l'époque contemporaine, puisque beaucoup plus d'objets ont été conservés. On retiendra l'exemple

**Fig. 1.4.** (a) Maquette du *Duomo* de Florence en Italie. (b) Le dôme tel qu'il a été construit. Les maquettes 3D, pourtant très onéreuses, ont néanmoins été utilisées depuis longtemps pour faciliter la compréhension de projets architecturaux jugés de grande importance : la 3D permet une compréhension intuitive bien plus facile qu'un plan.

frappant des plans d'architecte, en 2D, doublés pour des cas tout à fait exceptionnels de maquettes en 3D, pourtant extrêmement coûteuses. Et pourquoi la 3D ? Bien évidemment, comme toujours, parce que cela permet une compréhension infiniment plus facile de ce qui a été représenté.



(a) (b) (c) (d)

**Fig. 1.5.** (a) et (b) Cathédrale Sainte-Croix d'Orléans, 1601. (c) et (d) Cathédrale Notre-Dame d'Amiens, 1220. Dans la culture catholique, on trouve aussi de superbes développements géométriques, mais de façon annexe : la finalité essentielle des vitraux, c'est de narrer des histoires jugées importantes. Et les nombreuses statues présentes dans le porche d'entrée représentent une manière de rendre beaucoup moins distants (les vertus de la 3D !) des personnages clés de l'histoire de la religion.

Et entre temps, on peut noter combien les enjeux autour de l'image et de la sculpture ont été élevés, si on se rappelle certains interdits religieux : pas d'images de l'homme et pas de statues dans l'islam [2] (d'où une incroyable richesse de décorations à motifs géométriques, la

2. *La légitimité des images en islam : certes le Coran ne les défend pas ; il se contente de mettre en garde contre les idoles et par conséquent contre une idolâtrie éventuelle du prophète ou des saints. En revanche la tradition contenue dans les hadith insiste sur l'interdiction faite à l'artiste de représenter la vie, le monde tel qu'il est, pour ne pas mimer*

géométrie étant depuis toujours considérée comme d'essence divine), pas de représentations de Jésus et pas de statues dans le christianisme réformé, alors qu'au contraire le catholicisme utilise la statuaire comme un outil de communication, les portails des cathédrales gothiques et leurs vitraux jouant, à un coût très élevé justifié par l'enjeu, le rôle des bandes dessinées modernes.

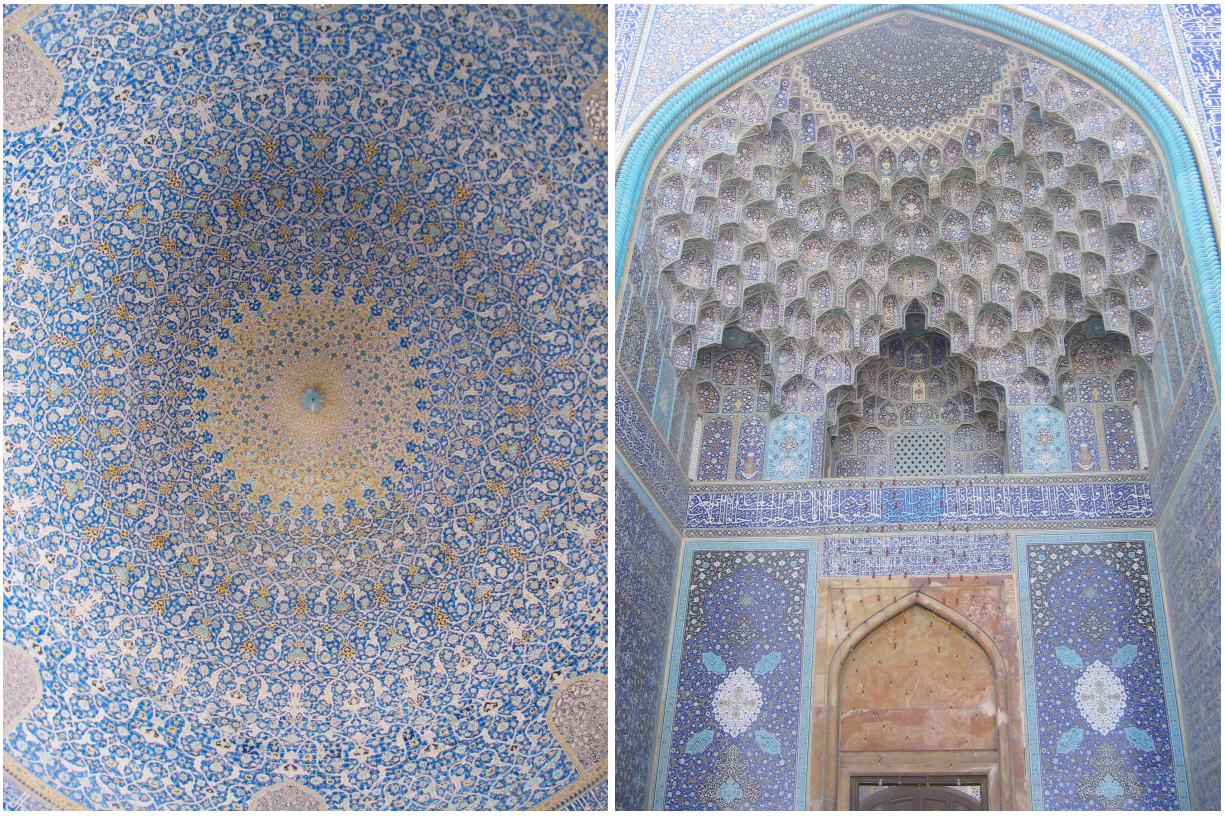


**Fig. 1.6.** La Mosquée du Shah, Isfahan, Iran, 1612. Dans la culture islamique, la richesse des développements graphiques purement géométriques a atteint de véritables sommets.

Nous venons de voir l'importance de la représentation 3D dans l'histoire, mais qu'en est-il aujourd'hui ?

Les champs d'utilisation de la 3D ont très clairement gagné une nouvelle partie du grand public. Certes, depuis l'invention de la photographie, il avait accès à des scènes distrayantes ou touristiques présentées en stéréo avec un matériel très peu onéreux. Mais les deux principaux obstacles à la généralisation de ce type de présentations n'ont disparu que très récemment. Le premier, c'est la puissance de calcul des ordinateurs, il n'est pas lieu de détailler ceci,

*le Créateur, et l'idéologie musulmane a volontiers incliné à l'abstrait, voire à l'iconoclasme. Cela a eu comme première conséquence d'exclure toute figure des textes coraniques et de ne rendre possible l'illustration de la vie de Mahomet que dans des époques de grande tolérance ou dans des milieux chiites libéraux. Par ailleurs, pour ne pas heurter les susceptibilités des croyants tout autant que par inclination personnelle, le peintre aura souci de ne pas traduire la réalité du monde sensible. Ce souci, remarquable dès les premières œuvres, ne cessera de s'affirmer davantage pour devenir absolu dans la miniature classique de l'Iran.* Extraits du texte : "La miniature iranienne, un art figuratif en terre d'islam" de Jean-Paul Roux

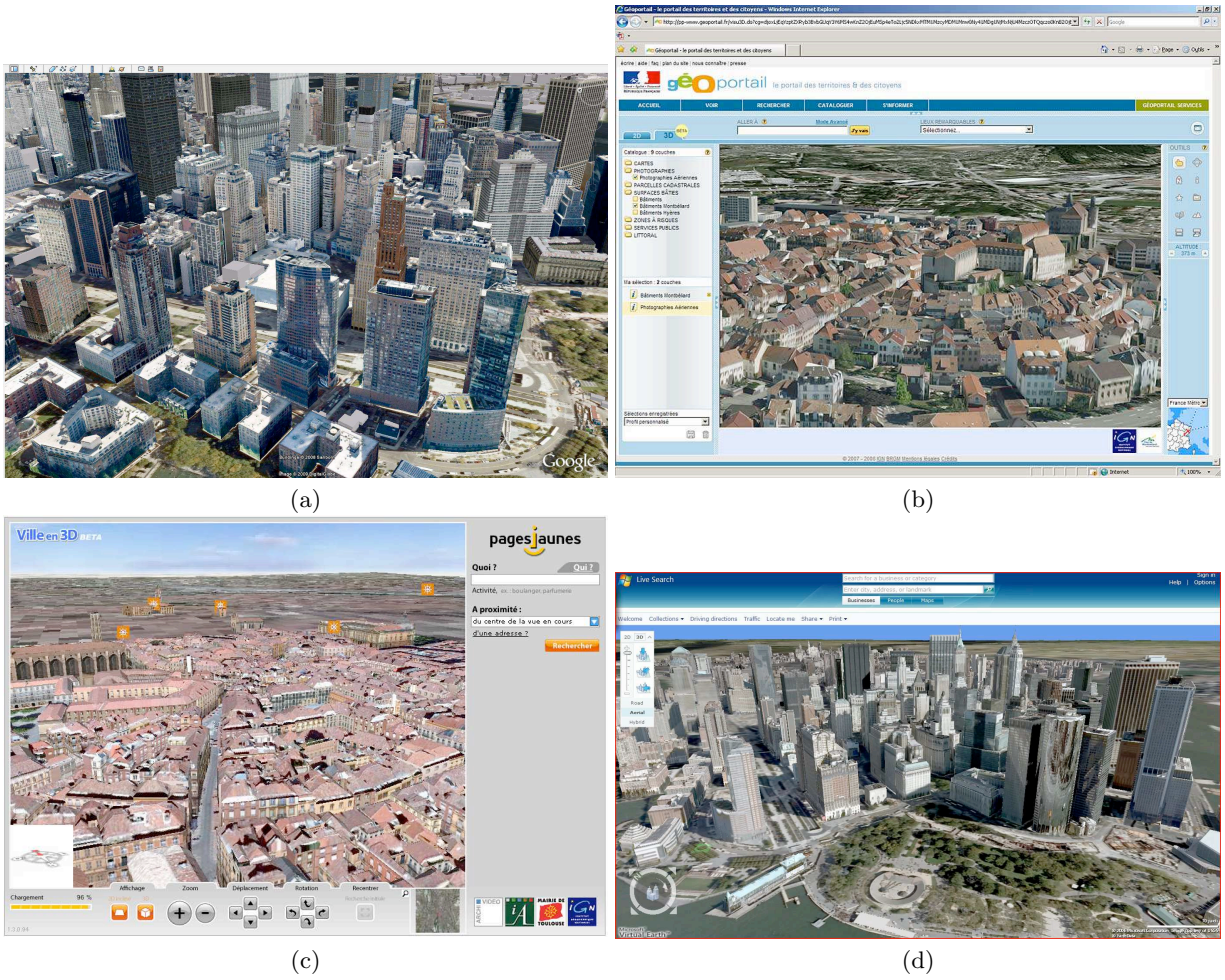
permettant de présenter du pseudo 3D sur des écrans courants. La scène fait alors l'objet de calculs temps réel, donnant à l'observateur la possibilité de se déplacer comme il le souhaite, ce qui restitue très exactement la sensation du relief avec un ordinateur sans matériel particulier. Le second, c'est la capacité de présenter du vrai 3D au grand public, films en relief (avec des lunettes polarisantes), écrans à micro-prismes, etc... toutes sortes de dispositifs qui permettent une véritable stéréoscopie avec un matériel pas très coûteux. Et dans le même registre, lorsqu'il faut quand même en arriver à fabriquer rapidement une maquette en 3D, on dispose même de moyens de la réaliser directement en sortie d'ordinateur, par des procédés de polymérisation sélective de zones très bien délimitées sous éclairage UV : le terme d'imprimante 3D est même souvent employé. En bref, la 3D s'est donc débarassée de l'obstacle majeur que représentait autrefois la fabrication pratique des statues, maquettes, etc., très gourmande en main d'oeuvre et donc très onéreuse.

Et puis, la reconstitution de scènes 3D en temps réel à partir d'images stéréoscopiques est elle aussi entrée dans le grand public. Ceci est apparu de façon très récente, avec la nouvelle génération de consoles de jeu qui analysent en temps réel avec une excellente précision la position des mains et du corps de chaque joueur et qui parviennent ainsi à se passer complètement d'organes de commande.

**Fig. 1.7.** Illustration de l'écran 3D WOW de Philips. Dans l'équipe Images Vidéo Communications de l'IRCCyN, cet écran est utilisé pour deux axes de recherches majeurs : la qualité d'expérience et l'étude de la fatigue visuelle. Pour la qualité d'expérience, les recherches se portent sur la définition de la notion de qualité pour les écrans 3D (différente de la 2D) et l'amélioration du rendu des différents écrans autostéréoscopiques. Pour la fatigue visuelle, l'équipe essaye de trouver et valider des protocoles afin de mesurer la fatigue visuelle engendrée par ces nouveaux médias.

Un nouvel épisode, absolument capital, de cette histoire est en train de se dérouler sous nos yeux. La géomatique a, comme beaucoup d'autres domaines, tiré le plein bénéfice de la montée en puissance des moyens de calculs individuels. Elle a permis le changement complet de la relation entre l'homme et la 3D, en démocratisant celle-ci dans des proportions jamais rencontrées auparavant.

La géomatique moderne connaît actuellement une phase d'expansion extrêmement importante, qui l'a conduite depuis quelques années à entrer dans le champ du grand public. Après le GPS dans les voitures et pour les randonnées, qui a amené beaucoup d'usagers à se frotter aux complexités de cartes au format électronique, c'est ensuite Google qui a donné le principal coup d'envoi de ce mouvement, il y a juste quelques années.




(a) (b)

(c) (d)

**Fig. 1.8.** (a) Google Earth. (b) Le Géoportail de l'IGN. (c) Les Pages Jaunes 3D. (d) MS Virtual Earth. Voici quelques unes des solutions récentes qui permettent un accès réellement facile et très peu coûteux à la 3D aujourd'hui : il s'agit d'une des principales révolutions vécues par la géomatique actuellement, qui répond à une attente qui vient du fond des âges : s'approprier l'image du monde sous une forme permettant une compréhension immédiate et spontanée.

Les "globes virtuels" se sont ensuite multipliés, et l'engouement du grand public pour ces données n'a cessé de croître, porté par les améliorations considérables des performances des

réseaux et de l'informatique personnelle. Plus récemment, cette géomatique grand public s'est emparée de la représentation 3D des bâtiments, puis de villes entières, support devenu normal pour des publicités, ou même outil au service des élus pour communiquer avec les citoyens. Le besoin d'acquérir des images et de les restituer en 3D, sous forme de maquettes parfaitement fidèles à la réalité, est ainsi devenu immense. On a donc vu, depuis une décennie, se construire des véhicules capables de photographier en stéréoscopie des villes entières, et il a fallu concevoir les algorithmes capables de traiter ces énormes quantités d'images. Très naturellement, les industriels en charge de ces problèmes se sont tournés vers les outils de vision par ordinateur et de robotique, très orientés vers le temps réel, oubliant l'essentiel de l'héritage de la photogrammétrie, orientée quant à elle vers une extrême précision, jugée ici comme une moindre priorité. Néanmoins, les algorithmes disponibles en vision par ordinateur présentaient de réels défauts lorsqu'ils étaient appliqués à des surfaces planes et, manque de chance, ce cas est extrêmement courant dans des scènes urbaines (les façades...).

Nous nous sommes intéressés à travailler sur des solutions nouvelles, capables d'exploiter les spécificités de telles images : tout d'abord, simplement pour accélérer l'orientation relative des images, en tirant bénéfice des points de fuite figurant dans celles-ci. De nouvelles méthodes d'extraction automatique de ces points ont été mises au point, bien plus performantes que celles disponibles jusqu'ici. Ensuite, nous avons souhaité corriger le défaut évoqué précédemment pour les surfaces planes, et nous avons élaboré de nouveaux algorithmes capables de donner en temps quasi-réel de bonnes solutions d'orientation relative, et ceci aussi pour de telles scènes. A cette fin, nous avons exploité de nouveaux outils mathématiques en rupture complète avec ceux utilisés traditionnellement. Finalement, nous avons essayé d'accélérer les méthodes d'orientation relative en exploitant opportunément la connaissance de la direction verticale, obtenue par exemple à l'aide de notre algorithme de détection des points de fuite.

Le présent travail est donc consacré de façon très complète à la remise à plat des solutions permettant l'orientation et la localisation de tout un ensemble d'images, et il a été très largement motivé par ce besoin de mieux exploiter les images destinées à fournir des maquettes très économiques de scènes urbaines.

## Plan du manuscrit

La première partie de ce travail (partie I) présente l'état de l'art sous deux facettes : imagerie et bases de GrÖbner.

Un historique de la photogrammétrie ainsi que de sa voisine plus jeune, la communauté de la vision par ordinateur, est présenté dans le chapitre 2. Il est en effet important de connaître les origines et les objectifs recherchés dans ces deux domaines pour une meilleure compréhension des méthodologies utilisés par la suite.

Le but principal est d'obtenir une troisième dimension à partir d'images à deux dimensions. Pour cela il faut arriver à calculer la position des caméras au moment des prises de vues, c'est-à-dire l'orientation et la localisation de chaque image dans l'espace. Nous détaillerons

alors les différentes étapes principales, depuis l'acquisition des images jusqu'au calcul d'un modèle 3D. Nous exposerons le point de vue de ces deux communautés dans leurs quêtes de cette fameuse 3D.

Depuis les origines et jusqu'à une période récente, toutes les opérations en photogrammétrie se faisaient de manière manuelle. Un opérateur devait regarder et détecter les points communs entre les différentes images, appelés points de liaison. Ces points sont nécessaires pour le calcul de l'orientation et de la localisation des images. Aujourd'hui, les avancées remarquables en traitement d'images ont permis une automatisation de ce processus de détection de points homologues. Nous discuterons des méthodes d'extraction et d'appariement de ces points dans la deuxième partie du chapitre 2, (paragraphe 2.3). L'extraction automatique de segments (qui sera utilisée par la suite) est aussi présentée dans ce chapitre.

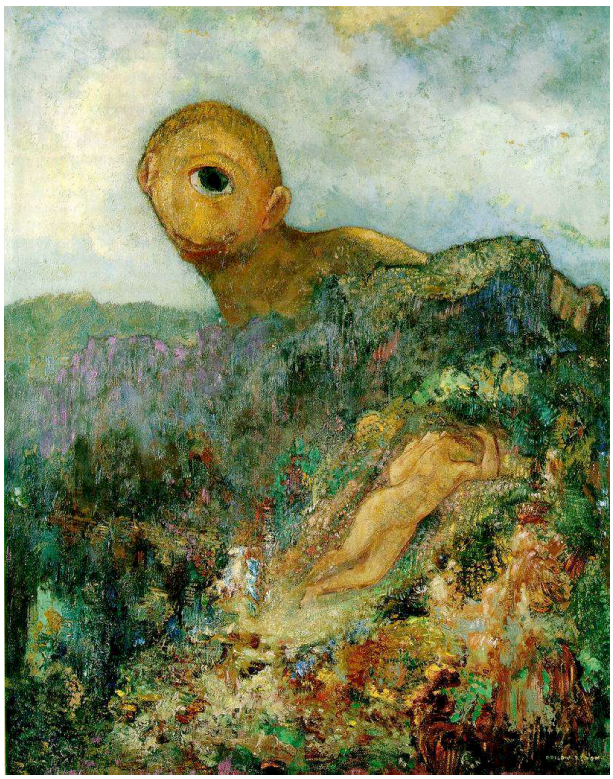


**Fig. 1.9.** Le cyclope par Odilon Redon, 1914, Tableau, huile sur bois, Museum Kroller-Mueller, Pays-Bas.

Pour clôturer cette partie, nous introduirons dans le chapitre 3, un outil mathématique récent appelé "bases de Gröbner". Notre objectif principal dans cette étude est de fournir de manière directe des solutions à des problèmes non-linéaires, qui traditionnellement se résolvent avec une linéarisation, ce qui présente l'inconvénient majeur de nécessiter des valeurs approchées. Or aujourd'hui, en utilisant les bases de Gröbner et en s'appuyant sur la puissance des nouveaux processeurs et les grandes capacités des mémoires vives, il est tout à fait possible de résoudre des équations non-linéaires de manière formelle, sans passer par les habituelles solutions purement numériques. Dans le chapitre 3, en premier nous rappellerons

quelques définitions de la géométrie algébrique, ensuite nous présenterons les bases de Gröbner et leur emploi dans la résolution des équations polynomiales.

Extraire des informations 3D à partir d'images 2D, et plus précisément à partir d'une seule image, est tout à fait possible grâce à la connaissance de la localisation du point de fuite sur l'image. Sinon, comment un cyclope percevrait la 3D dans son environnement [3] ?

Dans la géométrie conique caractéristique de la vision humaine ou de la photographie, les lignes parallèles de l'espace objet se traduisent dans chaque image par des faisceaux de droites qui concourent sur des points de fuite. A chaque point de fuite une direction 3D est associée.

**Fig. 1.10.** Illustration du point de fuite central. *Si-o-se Pol* ou 33 Pol (le pont aux 33 arches), Isfahan, Iran.

Inconsciemment quand on regarde par exemple la Figure 1.11, on se dit que le photographe a regardé vers le haut car l'image est en contreplongée. Avec la connaissance des points de

3. Plaisanterie mise à part, rappelons que beaucoup d'animaux, pourtant équipés de deux yeux, n'utilisent pas la stéréoscopie puisque leurs deux champs de vision n'ont pas de zone commune. Leur appréciation des distances est complexe, utilisant la parallaxe entre des images acquises successivement par le même oeil. Et par ailleurs leur cerveau, comme le notre, est un véritable système expert disposant d'une énorme bibliothèque d'objets : leur taille étant connue, ceci leur permet de calculer leur distance

fuite on peut quantifier l'orientation qu'a eu l'appareil photo au moment de l'acquisition.

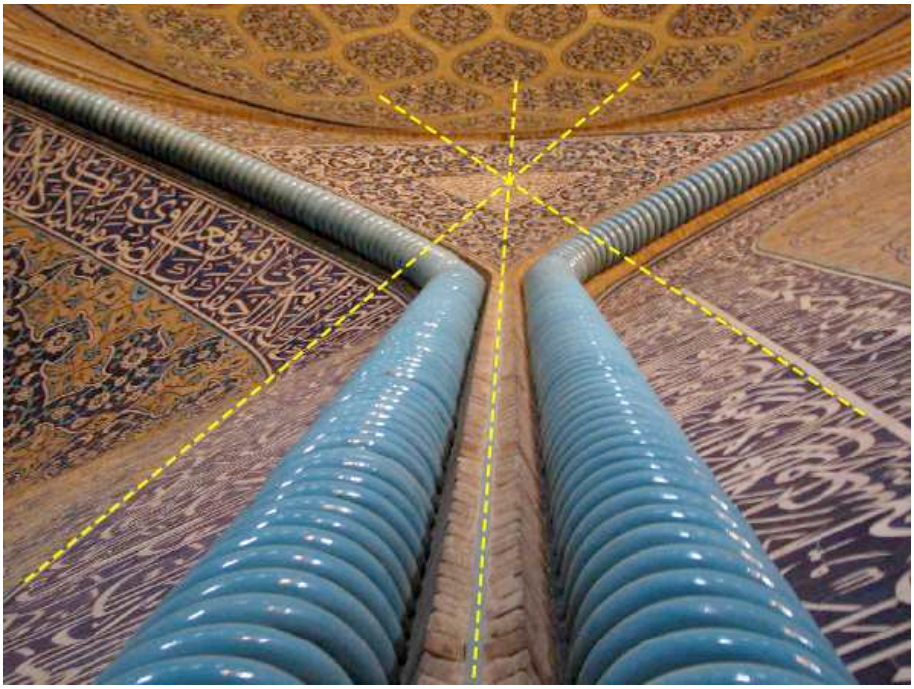


**Fig. 1.11.** Dans une image en contreplongée, le point de fuite des verticales est plus près du centre de l'image. Mosquée de Sheikh Lotf Allah, XVII$^{ème}$ siècle, Isfahan, Iran.

Dans la partie II, l'utilisation des points de fuite dans le calcul des orientations à partir d'une seule image est décrite. Après avoir introduit les points de fuite et l'état de l'art de leurs méthodes d'extraction, deux nouveaux algorithmes de détection sont présentés dans les chapitres 5 et 6. L'une de ces méthodes (chapitre 5) travaille dans l'espace de l'image, et réduit le problème de détection des points de fuite à une extraction de différents cercles. L'algorithme présenté dans le chapitre 6 quant à lui, travaille dans l'espace d'une sphère de rayon unité. Une comparaison des deux méthodes avec une méthode classique d'extraction de points de fuite est enfin présentée dans le chapitre 6.8.

Heureusement que nous avons deux yeux bien placés, ce qui nous permet de voir le monde en 3 dimensions, non comme ces malheureux cyclopes ! Nous allons donc utiliser deux prises de vue du même objet, et bien entendu il faudra que ces deux images, comme en vision humaine, se recouvrent de manière suffisante. Il est possible de pouvoir calculer la position et l'orientation de la deuxième image par rapport à la première même si aucune information extérieure n'est disponible, ce qui est appelé orientation relative. Une fois définis les éléments de l'orientation relative, nous pouvons obtenir un modèle 3D, qui à une échelle près formera une copie exacte de la réalité. L'emploi d'un système avec deux caméras est très fréquent en robotique, c'est ce qui permet au robot d'avoir une notion de la profondeur. On peut voir dans la Figure 1.12, la sonde Spirit du programme *Mars Exploration Rover* de la NASA [4].

4. http://marsrovers.jpl.nasa.gov

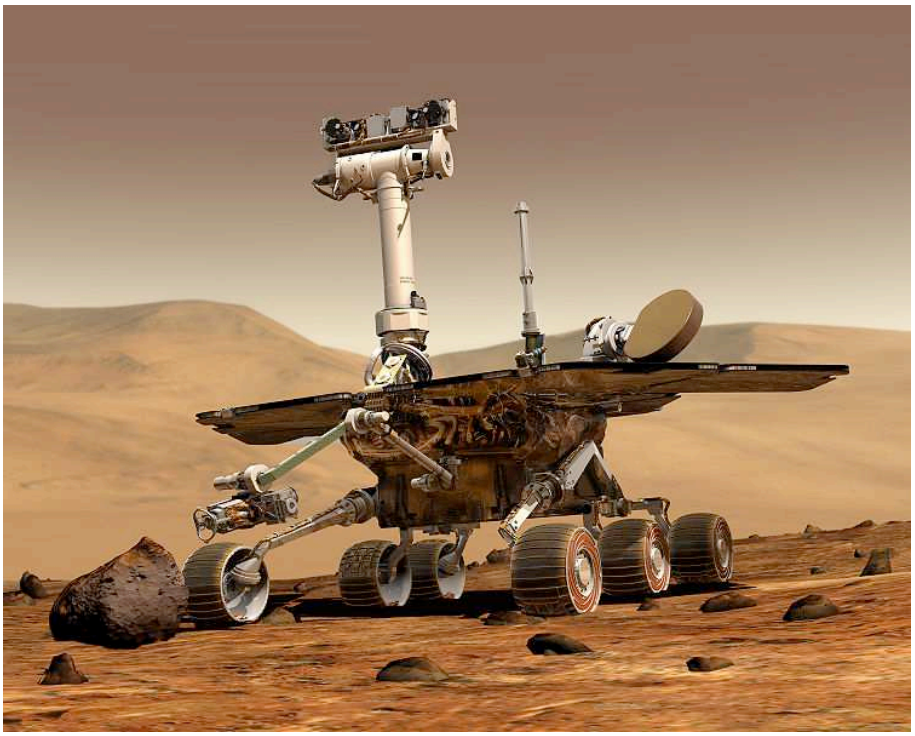


**Fig. 1.12.** Spirit, la première des deux sondes *Mars Exploration Rover* envoyées sur Mars en 2003, et qui ont atterri début 2004. Spirit est équipé de plusieurs caméras, dont une paire de caméras panoramiques (PanCam) de haute résolution, fixées au sommet du mât vertical porteur d'instruments. Chacun de ces instruments est équipé d'un capteur CCD de 1024 x 1024 pixels. Ce dispositif permet de restituer le relief, et de repérer les roches et les sols intéressants, en vue d'une analyse par les autres appareils de mesure.

Dans la partie III, nous travaillons sur un couple d'images stéréoscopiques. Une nouvelle formulation du problème de l'orientation relative à partir de 5 points homologues est présentée dans le chapitre 7. Cette modélisation calcule les inconnues de l'orientation relative sans passer par des calculs itératifs et surtout, sans besoin de valeurs approchées.

En introduisant la connaissance de la direction verticale, une information qui peut en particulier être donnée par les points de fuite, le problème de l'orientation relative devient plus simple et ne nécessite plus que 3 points homologues pour être résolu. Une nouvelle modélisation mathématique particulièrement simple est alors présentée dans le chapitre 8.

A ce stade nous avons seulement travaillé sur l'orientation d'un couple d'images, mais qu'en est-il pour plus de deux images ? Cette question fera l'objet de la dernière partie, la partie 9. Dans cette partie, en nous appuyant sur les méthodes existantes présentées dans le chapitre 2.2.3 de mise en place d'un ensemble d'images, nous présentons notre stratégie pour le calcul des orientations et des positions d'un ensemble complet d'images.

# Partie I

# Contexte méthodologique

# 2

# Introduction à la photogrammétrie et à la vision par ordinateur

## Sommaire



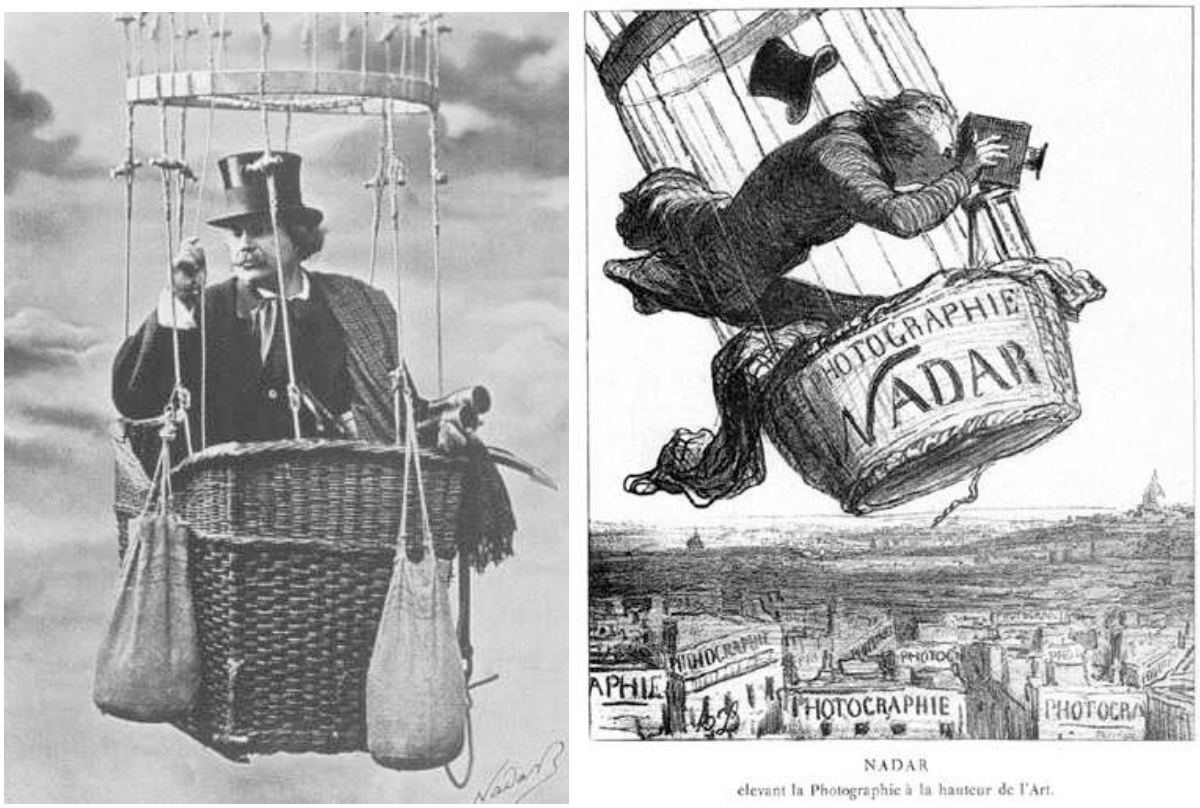
Dans ce chapitre, nous présentons dans un premier temps un bref historique de la photogrammétrie puis de la vision par ordinateur. Ensuite, nous passons en revue les différentes étapes importantes de l'obtention de mesures 3D à partir d'images pour ces deux communautés.

## 2.1 Historique

*La photogrammétrie*

La photogrammétrie est une technique qui a suivi très directement l'invention de la photographie au $XIX^{\text{ème}}$ siècle. L'exploitation d'images pour mesurer les distances de différents objets n'était conçue que comme une simple réutilisation de techniques de topographie, de type triangulation et intersection. Et comme les publications de ces techniques, dès le $XVI^{\text{ème}}$

siècle, le montraient bien, les applications envisagées étaient d'abord de type militaire : comment ajuster le tir d'un canon, comment cartographier une place forte ennemie sans s'en approcher, etc.




(a) (b)

**Fig. 2.1.** a) Nadar en aérostat. b) "Nadar", dessin de Honoré Daumier, 1862.

Dès que Nadar a produit, dès le milieu du $XIX^{\text{ème}}$ siècle, les premières images aériennes depuis un aérostat, ce sont des applications militaires qui ont encore été le moteur principal de la photogrammétrie naissante. A. Laussédat, l'inventeur de cette technique, était polytechnicien et officier du Génie. A cette époque, la cartographie nationale était elle aussi sous tutelle militaire, comme dans pratiquement dans tous les pays. Et donc très logiquement cette technique a été rapidement orientée vers des applications de cartographie, sur des surfaces très étendues, et ceci donc de façon de plus en plus industrielle. En parallèle, des applications architecturales ont elles aussi aussi été développées dès le début, mais sans rencontrer des débouchés commerciaux de même niveau, et ce ne sont donc pas elles qui sont devenues prépondérantes.

Ceci explique que la principale orientation de la photogrammétrie, quasiment dès sa naissance, a été la cartographie. Beaucoup de conséquences en découlent, en particulier :

1. une recherche de précision au meilleur niveau, afin de faire la meilleure cartographie possible avec un nombre minimal de photos, autrefois très onéreuses,

2. un travail avec un axe optique quasi-vertical (prises de vues aériennes), et donc des photos à axes presque parallèles,
3. un travail de restitution qui peut prendre beaucoup de temps, le délai entre la prise de vues et la cartographie résultante pouvant se chiffrer en mois, voire en années.
4. des développements qui progressivement ont quitté le domaine académique (avec de nombreuses publications), pour devenir quasi-exclusivement du domaine industriel, sans aucune publication. Ceci s'est traduit dans les dernières décennies, lors du passage au numérique, par de véritables boîtes noires sans aucun moyen pour l'usager de savoir ce que ces dernières contiennent. On a d'ailleurs assisté régulièrement à des travaux de recherche, dans des domaines connexes potentiellement usagers de cette technique, qui en ré-inventaient tout ou partie, par faute de publications accessibles. Quand une technique est ainsi portée par les seuls industriels, c'est un écueil fréquent, les étudiants cherchent des publications de recherche et n'en trouvent pas, alors qu'il s'agit de sciences de l'ingénieur, la partie récente étant presque totalement couverte par le secret industriel.

*La vision par ordinateur*

La vision par ordinateur est par contre un domaine qui n'a guère plus de trois décennies d'existence. Le domaine s'est développé dès qu'on a su numériser des images vidéo, et il couvre de nombreuses applications orientées vers le temps réel, ceci incluant l'extraction automatique d'éléments dans l'image. D'abord, simplement les contours, puis des éléments de plus en plus évolués, tels que des objets connus (des pièces mécaniques empilées en vrac), ceci allant jusqu'à des objets très complexes (reconnaissances de visages). Et puis ensuite, la volumétrie des objets visibles, à partir d'images prises de deux points de vues différents, et permettant un effet stéréoscopique. Une utilisation évidente a été le domaine de la robotique, l'objectif étant de permettre à une plateforme autonome de cartographier en temps réel son environnement immédiat, avec une exigence de précision assez modeste, mais variable : comme pour tout être vivant, le besoin de précision est d'autant plus grand que les objets sont proches, et la vision humaine est parfaitement adaptée à ce besoin.

Dans cette communauté règne une intense activité de recherche, poussée par des demandes industrielles très fortes, qui atteignent actuellement le grand public : citons par exemple, dans ce domaine, pour les appareils photos actuels les mises au point automatiques qui localisent d'elles-mêmes la zone d'image sur laquelle elle doivent s'exercer, ou même mieux, la photo qui ne se déclenche que quand le sujet photographié sourit et ne ferme pas les yeux : véritable prouesse, inaccessible il y a encore quelques années. Mais dans les domaines techniques professionnels, la multiplication des surveillances vidéo a contribué, à son tour, à susciter une demande considérable pour trouver de façon automatique, parmi des millions d'heures d'enregistrements, tel type d'objet, de véhicule, de visage, etc...

Donc la photogrammétrie et la vision par ordinateur partagent indiscutablement une même recherche de mesure 3D à partir d'images permettant la stéréoscopie, mais leurs passés respectifs et leurs clientèles très différentes les ont amenées à se développer de façon complètement parallèle, avec peu de zones communes.

## 2.2 Les principales étapes de la photogrammétrie et vision par ordinateur

Dans cette partie les différentes étapes de l'orientation 3D d'images en photogrammétrie et son analogie en vision par ordinateur sont détaillées. Elles sont illustrées dans la Figure 2.2.

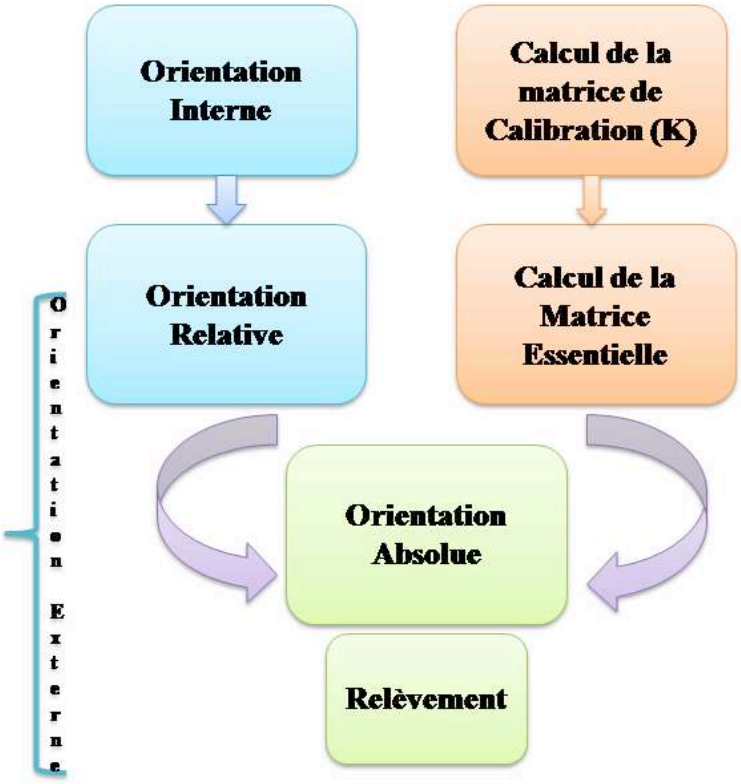



**Fig. 2.2.** Les principales étapes en photogrammétrie (les blocs bleus) et vision par ordinateur (les blocs oranges)

Comme on peut voir dans la Figure 2.2, la première étape est celle du calcul des paramètres internes, ce qu'on appelle en photogrammétrie l'orientation interne, et en vision par ordinateur, le calcul de la matrice de calibration. Ensuite, cette étape est suivie du calcul des paramètres d'orientation et de localisation d'une image par rapport à une autre. Cette étape est appelée orientation relative en photogrammétrie. Et en vision par ordinateur, à ce stade on calcule la matrice essentielle, ce qui est équivalent à l'orientation relative.

Après cette étape, les images sont mises en place dans l'espace dans un référentiel quelconque, en général on prend le sommet de prise de vue de l'image de gauche comme origine du système de coordonnés terrain. Le modèle est, par la force des choses, sans échelle.

Dans beaucoup d'applications on peut envisager de s'arrêter là, mais si on veut faire de la cartographie, ou bien avoir de vraies mesures, il est indispensable de pouvoir mettre dans un

système de référence connu les images qui ont été mises en place dans un système quelconque. Cette dernière étape est appelée orientation absolue.

Dans les prochains paragraphes, nous allons détailler toutes ces étapes.

### 2.2.1 L'orientation interne

La première chose à définir est le modèle de la caméra. Par défaut on utilise comme modèle nominal celui d'une caméra dite à perspective centrale (*pinhole camera*). Tous les rayons passent par le même point, nommé centre optique ($C$). Derrière ce point, à la distance focale ($F$), se trouve le plan image.

L'intersection de l'axe optique avec le plan image donne le *point principal de symétrie, PPS*. Le *PPS* a une réalité physique. Par ailleurs, en général le plan image n'est jamais vraiment perpendiculaire à l'axe optique, et il existe nécessairement un défaut d'orthogonalité, que les constructeurs gardent à une valeur aussi faible que possible.

Pour des raisons de simplification de la modélisation, et donc des calculs, un point théorique au nom de *point principal d'autocollimation, PPA* est défini. Ce point, comme évoqué par le terme d'autocollimation, est le pied de la perpendiculaire sur le plan image du vecteur reliant le centre optique à l'image, voir Figure 2.4.

Le système d'axes image qui a comme origine le PPA avec l'axe des Y vers le haut est appelé repère photogrammétrique, voir Figure 2.3.

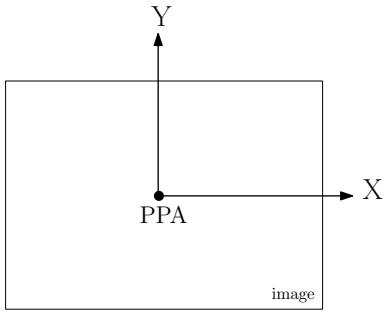



**Fig. 2.3.** Illustration du repère photogrammétrique

En général, quand on parle du centre de l'image, c'est à ce point qu'on fait allusion.

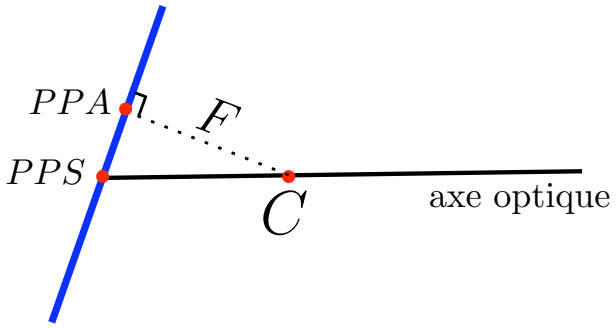



**Fig. 2.4.** Illustration montrant les positions du PPA et du PPS

La Figure 2.4, est un schéma très simplifié de la géométrie réelle, ce schéma est correct si les conditions de l'approximation de Gauss sont respectées. Rappelons que l'approximation de Gauss (d'après le physicien allemand Carl Friedrich Gauss) est une approximation de l'optique géométrique, correspondant à un cas où les angles d'incidence des rayons sont faibles et où le point d'incidence est proche de l'axe optique.

Les écarts à cette approximation engendrent les aberrations géométriques, d'inclinaison (aberrations qui augmentent avec l'inclinaison du rayon par rapport à l'axe optique), et/ou d'ouverture (qui augmentent avec la distance entre le point d'entrée du rayon et le centre optique).

Au cours des dernières décennies, dans l'industrie de l'optique, de nombreuses améliorations concernant les lentilles sont intervenues afin de réduire les effets de la plupart des aberrations. Les lentilles de forme traditionnelle sont de forme sphérique, seule forme facile à polir depuis les premiers quatre siècles d'existence de cette technologie, ce qui est l'origine de ces aberrations optiques. Par exemple, les rayons qui passent par le centre ne convergent pas tout-à-fait au même point que ceux qui passent par les bords, ce qui provoque en particulier un effet de flou, surtout aux grandes ouvertures.

Mais depuis quelques années on a pu noter un emploi massif de lentilles moulées asphériques. Comme leur nom l'indique, ce sont des lentilles qui sont fabriquées par réplication ultra précise d'un moule épousant une forme différant légèrement d'une sphère, ce qui enlève une grande partie des aberrations géométriques. Avec les lentilles sphériques traditionnelles, afin de pouvoir corriger une partie des aberrations géométriques, il fallait en combiner un grand nombre, impliquant une forte perte de luminosité. En plaçant les lentilles asphériques dans des endroits stratégiques de l'objectif, on peut enlever une grande partie de ces mêmes aberrations géométriques, et ceci tout en améliorant la luminosité de l'image grâce à la diminution du nombre de lentilles qui en résulte, sans même évoquer les gains issus des traitements anti-reflet modernes.

On peut voir dans la Figure 2.5, comment une lentille asphérique fait en sorte que tout les rayons convergent en un même point.

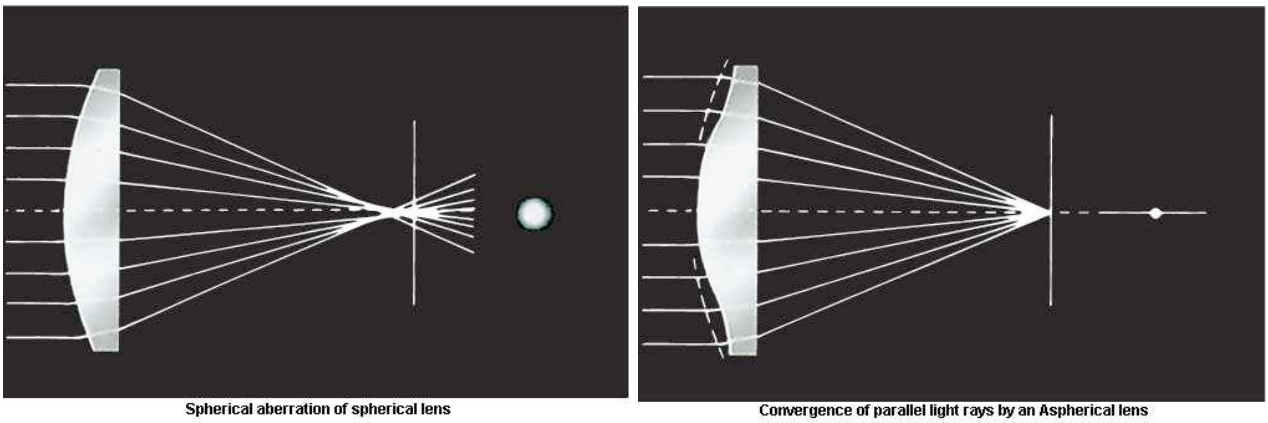



**Fig. 2.5.** Illustration de correction du flou avec l'aide d'une lentille asphérique. a) Une lentille classique sphérique, b) une lentille asphérique. Image tirées du site de CANON

Dans la Figure 2.6, est présenté un exemple d'objectif CANON EF-S 17-85mm f/4-5.6 IS USM [1] et son diagramme de lentilles. On peut voir l'emplacement de la lentille asphérique en vert.

**Fig. 2.6.** Exemple d'un objectif CANON (EF-S 17-85mm f/4-5.6 IS USM) et de l'emplacement de sa lentille asphérique. Image tirée du site de CANON

La seule aberration qui présente un caractère peu gênant pour le grand public est la distorsion, aberration due à la seule inclinaison. Par contre, afin d'obtenir des résultats précis pour un usage en photogrammétrie et en vision par ordinateur, il est indispensable de savoir modéliser les effets de cette distorsion. A titre indicatif dans de nombreux appareils photo grand public, elle peut atteindre jusqu'à 40 pixels sur les bords, il est donc exclus de l'ignorer alors même que les résidus de compensation sur les points d'appui sont fréquemment inférieurs à 0.1 pixel. La distorsion est radiale, elle présente une symétrie de révolution par rapport à l'axe optique. La distorsion est définie selon le modèle des aberrations de Seidel [Meyzonnette, 2003], par le polynôme suivant, nécessairement impair, et dont l'ordre 1 est nul lorsque la focale est bien déterminée : $dr = ar^3 + br^5 + cr^7$ $r$ étant la distance radiale depuis le $PPS$ dans le plan image et $a$, $b$, et $c$ les premiers coefficients de cette distorsion

1. http://www.canon.com/

radiale (d'expérience, ces trois premiers coefficients sont suffisants pour bien la modéliser).

**Fig. 2.7.** Exemple d'une image avec distorsion, très nettement visible sur le haut du bâtiment (bâtiment de l'ESGT, Le Mans).

L'étape de calcul des trois coefficients de la distorsion ($a$, $b$, $c$), le $PPS$, la distance focale ($F$) et le $PPA$, est nommée *orientation interne* en photogrammétrie. Nous ne rentrerons pas dans les détails de ce calcul. Ces paramètres évoqués précédemment sont appelés généralement les paramètres intrinsèques de l'image.

Le processus de calcul des paramètres internes de la caméra est appelé étalonnage.

En vision par ordinateur, chaque point sur l'image est représenté avec une matrice 3 x 1 avec une représentation homogène, $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.

Les paramètres de l'orientation interne sont exprimés sous forme d'une matrice 3 x 3, appelée matrice de calibration ($K$) avec $K = \begin{bmatrix} F_1 & \gamma & x_{ppa} \\ 0 & F_2 & y_{ppa} \\ 0 & 0 & 1 \end{bmatrix}$ $F_1$ et $F_2$ représentent la distance focale selon les deux directions $X$ et $Y$, modélisation classique en vision par ordinateur qui permet de traiter le cas où les pixels ne sont pas parfaitement carrés. En général une même valeur est prise pour les deux. $x_{PPA}$ et $y_{PPA}$ sont les coordonnées image du $PPA$. $\gamma$ est le facteur de déviation, en général nul, qui permet de prendre en compte d'autres formes de défauts. Quand on dispose de cette matrice $K$ en vision par ordinateur, on parle alors

d'images étalonnées, *calibrated images*, souvent traduit improprement par "images calibrées". La matrice de calibration ($K$) est appliquée à tous les points images exprimés en coordonnés pixel.

Dans notre étude nous avons utilisé l'appareil photo reflex Nikon D70s (voir Figure 2.8) avec un objectif de focale fixe de 20 mm.


**Fig. 2.8.** Appareil photo Nikon reflex D70s

Le graphe de la distorsion est présenté dans la Figure 2.9.

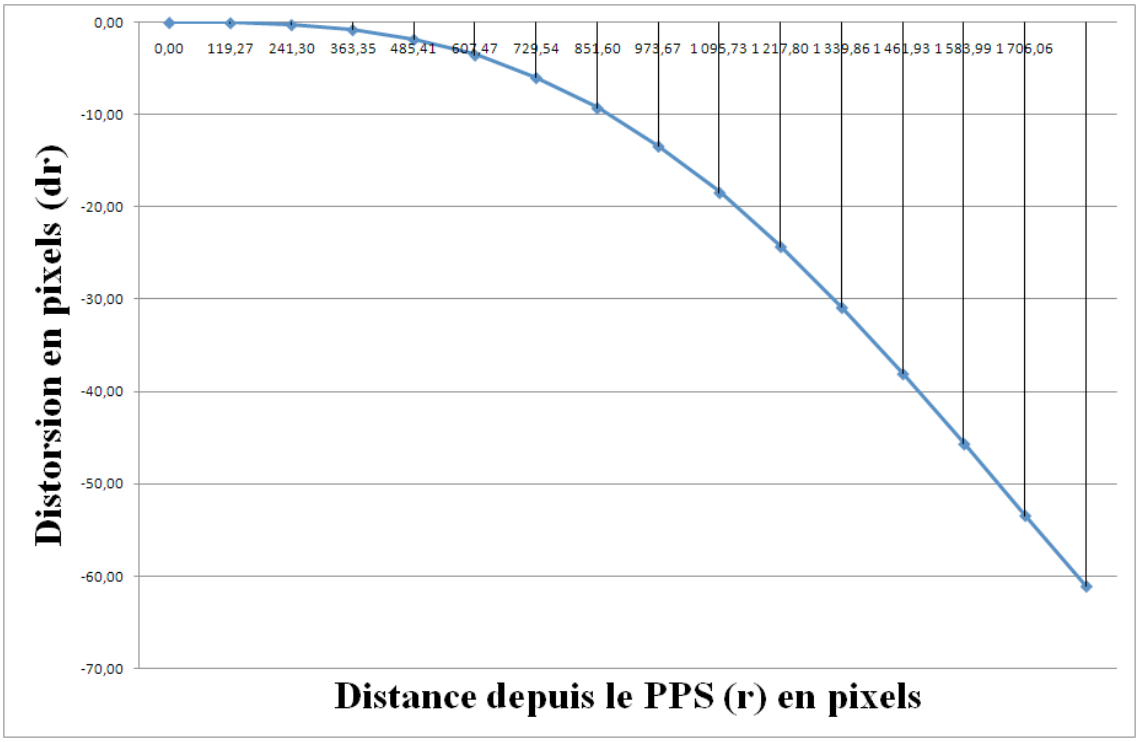



**Fig. 2.9.** Distorsion typique d'un objectif d'appareil numérique courant

Comme on peut le constater sur les bords nous avons une distorsion de 60 pixels. Dans la suite de ces travaux, nous corrigeons donc de la distorsion toutes les images avant de les traiter.

### 2.2.2 L'orientation relative

Le calcul des paramètres extrinsèques (orientation et position) de la caméra à partir de plusieurs vues à l'aide des points d'intérêt est à la base de pratiquement toutes les applications aussi bien du domaine de la photogrammétrie que de celui de la vision par ordinateur. Bien que le problème soit le même pour ces deux communautés, chacune d'elles a proposé une approche différente afin de calculer la position et la localisation des caméras au moment des acquisitions de prises de vues.

Dans ce paragraphe, nous passons en revue les différentes méthodes existantes pour la résolution de l'orientation relative.

Une étape importante, pour ces deux communautés, est la mise en place des images dans l'espace, c'est-à-dire le calcul de la position et l'orientation des images au moment des prises de vues. En photogrammétrie cette étape est appelée l'orientation externe, en symétrique de l'orientation interne qui cherche à déterminer les paramètres de calibration de la caméra comme la focale, le centre principal d'autocollimation, le polynôme de distorsion ainsi que le centre principal de symétrie. Une première différence majeure entre la communauté de vision par ordinateur et photogrammétrie est dans le signe de la distance focale. En photogrammétrie, la focale a un signe négatif, ce qui correspond à la géométrie intuitive du modèle. En effet le rayon atteint d'abord le centre optique pour ensuite couper le plan focal. En revanche, dans toutes les modélisations de vision par ordinateur, la distance focale est positive, fondamentalement, ceci ne change rien aux équations.

L'orientation externe est basée sur les équations de colinéarité. Comme le nom l'indique, celles-ci consistent à dire que le point terrain $M$, le sommet de prise de vue $S$ (qui correspond au centre optique), et $m$ la projection de $A$ sur l'image, sont sur la même droite (Figure 2.10).

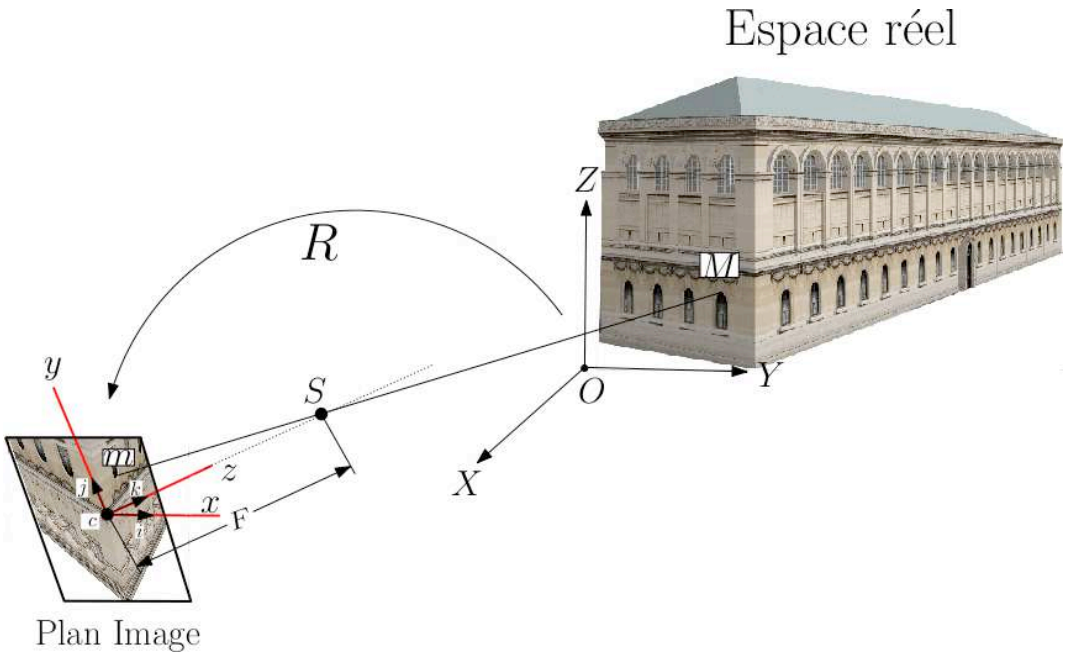



**Fig. 2.10.** La projection conique utilisée en photographie, formalisée par l'équation de colinéarité (illustrée par un modèle 3D de la bibliothèque Ste Geneviève, réalisé et mis à disposition par Leonhard Pröttel).

En entrée de l'équation de colinéarité, on a comme paramètres les coordonnées images des points et la calibration de la caméra, un point terrain étant nécessairement vu au moins sur deux images. Certains des points terrain sont exprimés dans un référentiel connu (les points d'appui), ce qui permet de réaliser l'orientation absolue. Mais on peut très bien dans un premier temps ne pas disposer de points d'appui, auquel cas on travaille dans un espace euclidien, mais sans échelle (orientation relative). En sortie de l'équation de colinéarité, nous pouvons obtenir l'orientation des images, les sommets de prises de vues, ainsi que les coordonnées terrain des points de liaison. Ces coordonnées sont obtenues soit dans un référentiel connu et avec une échelle, si nous avons fait l'orientation absolue, soit dans le référentiel de la première image et sans échelle si seule l'orientation relative a été effectuée. Il est tout à fait possible, à tout moment, de faire basculer les points 3D, obtenus à partir de l'orientation relative, dans un référentiel connu, et pour cela donc il faut disposer de points connus dans le référentiel voulu.

Revenons maintenant à la résolution de ces équations de colinéarité. Il faut au minimum 5 couples de points homologues pour pouvoir effectuer l'orientation relative, or en général bien plus de points peuvent être disponibles (par exemple en utilisant des outils d'extraction automatique de points d'intérêt, cf chapitre 2.3.1). Dans ce cas il faudra minimiser (par moindres carrés) le carré de la distance entre le point théorique terrain et celui que l'on calcule.

L'inconvénient majeur de l'approche photogrammétrique, qui est basée sur la condition de colinéarité, est du à la non-linéarité du problème ainsi posé, la non-linéarité des foonctions à minimiser impose la nécessité de disposer de valeurs approchées autour desquelles on opère

une linéarisation de la fonction analytique à minimiser.

En photogrammétrie aérienne [Kasser et Egels, 2001], [Wolf et Dewitt, 2000], [Moffitt et Mikhail, 1980], les valeurs initiales sont très simples à estimer en raison de la régularité d'échantillonnage spatial et des axes quasi-verticaux des clichés. En plus, l'avion ne subit que de faibles variations de rotation, et un ensemble de points d'appui peut toujours être obtenu.

Le problème est plus compliqué quand la photogrammétrie s'attaque à des prises de vues terrestres. La première grande différence est que les prises de vues ne sont plus à axes quasi parallèles, mais deviennent parfois franchement convergentes. D'autre part, à moins d'avoir des points d'appui connus, le calcul des paramètres approchés est lui aussi bien plus compliqué qu'en photogrammétrie aérienne.

Ce qu'on peut retenir ici est que la photogrammétrie, dans son utilisation classique sur vues aériennes, travaille dans une configuration significativement plus simple pour la résolution des équations de colinéarité. Et par la suite, il en est de même lors de la compensation par faisceaux, traitement qui s'applique lorsqu'on traite en bloc un grand nombre d'images.

Dans le contexte de vision par ordinateur et de robotique des années 80, il y avait ce besoin de pouvoir déterminer de manière directe et surtout linéaire les paramètres d'orientation et de position. Pensons à un robot qui doit se déplacer et voir le monde en trois dimensions pour obtenir des informations de type topologique, du genre "la table est derrière la chaise". Premièrement il n'est pas nécessaire pour ce genre d'application d'avoir des points d'appui, et ensuite une grande précision telle que celle que l'on cherche en photogrammétrie n'est pas nécessaire.

C'est pour cela qu'une nouvelle modélisation a été proposée, basée non pas sur les équations de colinéarité, mais sur une autre contrainte très simple elle aussi, appelée contrainte de coplanarité.

Comme on le voit mieux sur la Figure 2.11, la condition de coplanarité entre deux images exprime le fait que le vecteur de visée depuis le premier sommet de prise de vues $\overrightarrow{V_1}$, le vecteur de visée depuis le deuxième sommet de prise de vues (et exprimé dans le référentiel du premier) $\overrightarrow{V_2}$, ainsi que le vecteur de la translation (entre les deux sommets de prises de vues) $\overrightarrow{T}$ se trouvent dans le même plan, appelé le plan épipolaire.

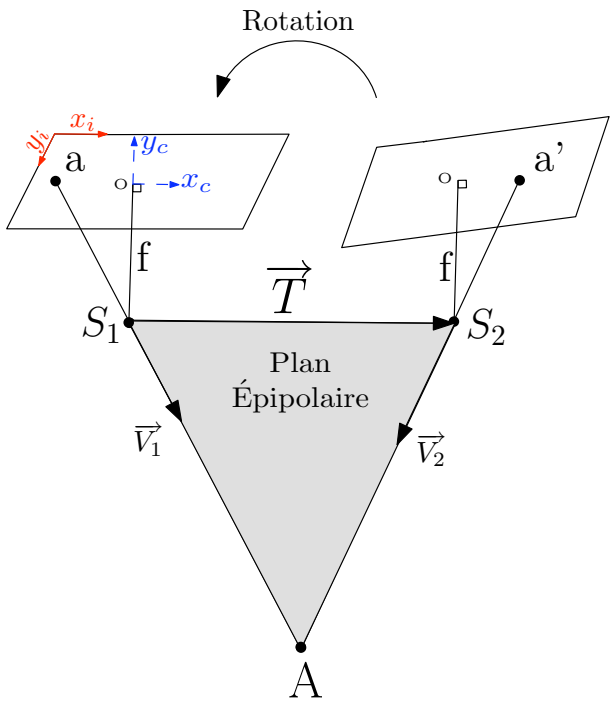



**Fig. 2.11.** La matrice rotation de la seconde caméra par rapport à la première est appelée R, et le vecteur de translation $T$ est la base qui relie les centres optiques des caméras ($S_1$ et $S_2$). $O$ est le point principal d'autocollimation (PPA). Les images du point terrain A sur les 2 images sont a et a'.

On peut traduire cette condition par un produit mixte nul entre ces 3 vecteurs. En d'autres termes :

$$\overrightarrow{V_2} \cdot (R\overrightarrow{V_1} \wedge \overrightarrow{T}) = 0. \tag{2.1}$$

Avec sa forme exprimée de manière algébrique :

$$\begin{bmatrix} x_{a'} & y_{a'} & f \end{bmatrix} \begin{bmatrix} 0 & T_z & -T_y \\ -T_z & 0 & T_x \\ T_y & -T_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ f \end{bmatrix} = 0. \tag{2.2}$$

La contrainte de coplanarité a été durablement exploitée par la communauté de vision par ordinateur. [Longuet-Higgins, 1981] publia la première fois en 1981 la notion de matrice dite "essentielle" ($E$), $E$ n'étant en fait que la multiplication de deux matrices :

$$\begin{bmatrix} 0 & T_z & -T_y \\ -T_z & 0 & T_x \\ T_y & -T_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}.$$

En utilisant la matrice essentielle ($E$) la condition de coplanarité s'exprime désormais de façon linéaire, et $E$ est une matrice de rang de 2. Résoudre l'orientation relative revient donc à estimer les 9 éléments de la matrice essentielle. En fait 8 points suffisent pour résoudre le système, car l'estimation se fait nécessairement à un facteur d'échelle près, et par ailleurs l'emploi de la matrice essentielle suppose que la calibration de la caméra est disponible. Cette résolution, bien qu'efficace en termes de calcul, ajoute 3 degrés de liberté de plus à l'orientation relative puisque, comme vu précédemment, l'orientation relative n'a nominalement que 5 degrés de liberté. Un autre inconvénient, majeur dans bien des cas d'applications en

zones urbaines, est qu'elle ne peut fonctionner si tous les points homologues sont coplanaires dans l'espace objet, ce qui peut très bien arriver par exemple sur des façades. Beaucoup de recherches au sein de la communauté de vision par ordinateur ont donc été menées pour réduire le degré de liberté de cette matrice essentielle, afin de réduire au strict minimum de 5 le nombre de couples de points homologues requis. D'ailleurs les différentes méthodes de résolution de la matrice essentielle ont pour nom ce nombre de points homologues minimum. On peut ainsi citer les méthodes des 8 points [Longuet-Higgins, 1981], [Hartley et Zisserman, 2004], [Ma *et al.*, 2003], 7 points [Sturm, 1869], 6 points [Hartley et Dano, 2000] et plus récemment 5 points, le strict minimum.

Il a été pour la première fois démontré par Kruppa [Kruppa, 1913] en 1913 que la résolution directe de l'orientation relative à partir de 5 points contenait en général 11 solutions au plus. La méthode décrite par celui -ci consistait à trouver toutes les intersections de deux courbes de degré 6. Malheureusement en son temps sa méthode n'a pu donner lieu à une implémentation numérique. Plus récemment dans [Demazure, 1988], [Faugeras, 1993], [Faugeras et Maybank, 1990], [Maybank, 1992], [Heyden et Sparr, 1999] il a été démontré que le nombre de solutions est en général égal à 10, incluant les solutions complexes. Triggs [Triggs, 2000] a donné une version détaillée pour une implémentation numérique. Philip [Philip, 1996] a présenté en 1996 une solution utilisant un polynôme de degré 13, et donné par ailleurs une méthode numérique pour résoudre son système, les racines de son polynôme donnant donc directement l'orientation relative. L'idée de Philip a été reprise en 2004 par Nister [Nistér, 2004] qui a affiné cet algorithme. Depuis, nombre d'articles ont essayé de donner des améliorations à la méthode de Nister, notamment [Stewénius *et al.*, 2006] qui a donné une résolution polynomiale à l'aide des bases de Gröbner. D'autres articles ont proposé quelques modifications de la méthode de Nister en vue d'une amélioration numérique, ou bien d'une simplification d'implémentation [Li et Hartley, 2006], [Batra *et al.*, 2007], [Segvic *et al.*, 2007] et [Kukelova *et al.*, 2008].

Aujourd'hui toutes les résolutions de l'orientation relative citées ici sont basées sur les propriétés de la matrice essentielle. La plupart des articles portent sur des tentatives d'amélioration de la stabilité numérique des résolutions polynomiale [Byröd *et al.*, 2009]. Car le fait que la rotation et la translation soient entremêlés engendre des instabilités, surtout quand par exemple la translation est très petite.

### 2.2.3 Calcul de l'orientation et de la position d'un ensemble d'images

#### Orientation absolue

Dans l'étape de l'orientation relative le système de référence est local, et son origine est traditionnellement basée sur le sommet de prise de vue de l'image de gauche. En photogrammétrie aérienne, dont la principale finalité est la fabrication de cartes et d'orthoimages, il est évidemment nécessaire de georéférencer toutes les images. Cette étape de basculement d'un repère local à un repère terrain connu est appelé orientation absolue [Kraus et Waldaeusl, 1998], [Grussenmeyer, 2002]. Les éléments à déterminer dans l'orientation absolue sont le facteur d'échelle, la translation et la rotation qui sont dépeints sur la Figure 9.3.

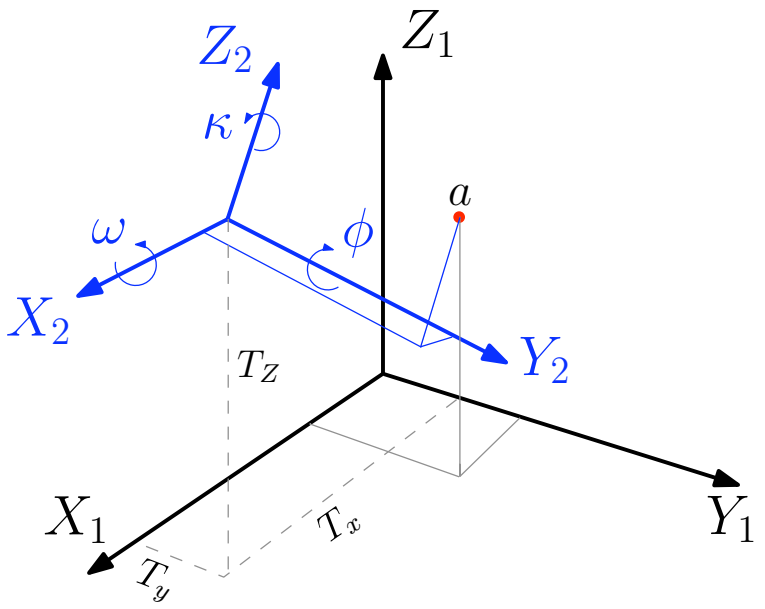



**Fig. 2.12.** Illustration de l'orentation absolue

[Horn, 1987] dans son article présente une méthode de résolution directe avec l'aide des quaternions.

### Le relèvement

Une autre stratégies adoptées par de nombreux chercheurs est de procéder dans un premier temps à l'orientation relative du couple d'images qui a le plus grand nombre de points appariés. Le résultat de cette étape est un ensemble de points en 3 dimensions, dans le référentiel de l'image de gauche de ce couple. Ceci revient à dire que le système d'axes du terrain est centré sur le sommet de prise de vue de l'image de gauche. Ces points terrain prennent à leur tour le rôle de points d'appui pour les autres images. Ensuite, pour chaque image que l'on souhaite rajouter, une recherche est faite afin d'apparier ses points d'intérêts avec le nuage de points 3D issu du premier couple. Le but est donc de pouvoir définir le sommet de prise de vue de cette image par rapport au référentiel terrain.

C'est un problème bien connu sous le nom de "relèvement" (en anglais : *space resection*) en topographie et photogrammétrie [Slama, 1980].

La littérature sur la résolution du relèvement est très riche, car outre les méthodes de résolution existantes en photogrammétrie, il y a énormément d'articles qui traitent le sujet dans la communauté de vision par ordinateur. Dans cette communauté le relèvement est connu sous le nom de *N points perpective pose estimation*.

La différence majeure entre ces méthodes est à chercher dans la linéarité ou non de celles-ci.

Une des modélisations les plus employées, compte tenu de sa linéarité et donc de sa simplicité de calcul, est celle de [Abdel-Aziz et Karara, 1971], [Marzan et Karara, 1975], et

elle est connue sous le nom de DLT *(Direct Linear Transformation)*. Cette approche a été développée à l'université de l'Illinois en 1971. L'orientation linéaire directe permet aussi de passer directement des coordonnées pixel de points homologues à leurs coordonnées terrain (équation 2.3). D'ailleurs il est possible de rajouter les paramètres de calibration, traités comme inconnues, aux équations 2.3 de DLT . Ceci permet alors l'étalonnage de la caméra.

$$\begin{aligned} x &= \frac{a_1X + a_2Y + a_3Z + a_4}{c_1X + c_2Y + c_3Z + 1}, \\ y &= \frac{b_1X + b_2Y + b_3Z + b_4}{c_1X + c_2Y + c_3Z + 1}. \end{aligned} \tag{2.3}$$

Les équations de DLT sont linéaires et contiennent 11 inconnues, plus 5 si l'on rajoute les paramètres de la calibration. Donc pour résoudre ces équations il faut au minimum 6 points (soit 12 coordonnées image) si la calibration est connue, ou 8 points dans le cas contraire.

Cette modélisation a été récemment employée dans Photosynth[2] le logiciel expérimental de Microsoft [Snavely *et al.*, 2008], voir Figure 2.13.


**Fig. 2.13.** Illustration tirée du site Web de PhotoSynth

[Brown et Lowe, 2005] emploie aussi cette stratégie de rajout des images, une par une, à un couple d'images initiales. Il procède ensuite à une compensation par faisceaux, voir figure 2.14.

2. http://photosynth.net/

(a) Input images

(b) Output 3D model 1 - Tiger

**Fig. 2.14.** Illustration tirée de l'article de [Brown et Lowe, 2005]

[Brown et Lowe, 2005] et [Snavely *et al.*, 2008] utilisent la méthode SIFT comme détecteur de points d'intérêts. Avant de commencer les calculs, ils classent d'abord les images qui ont un recouvrement, ce qui permet de traiter de grosses bases de données d'images.

Lorsqu'on connaît les paramètres de calibration de l'image, seulement 3 points terrain et leurs correspondances sur l'image suffisent pour calculer la position et l'orientation de l'image.

Les méthodes de relèvement ont fait l'objet de nombreuses publications depuis au moins deux siècles. [Ameller *et al.*, 2002] en attribue la première résolution à Joseph Louis Lagrange en 1795. Mais sa résolution de manière directe et rigoureuse à été faite pour la première fois par un mathématicien allemand du nom de Grunert [Grunert, 1841]. Sa solution a été ensuite raffinée en 1925 par le photogrammètre allemand Muller [Muller, 1925]. En 1949, [Merritt, 1949] donna de manière indépendante une solution au problème du relèvement direct.

L'importance du problème de la résolution directe du relèvement en photogrammétrie aérienne n'est pas très grande, étant donné qu'il n'y a pas de contraintes fortes au niveau du temps de calcul, et donc les méthodes itératives de résolution y sont faciles d'emploi. La première résolution itérative a été publiée la première fois par [Church, 1945], mais son algo-

rithme nécessite de très bonnes valeurs initiales. Celles-ci peuvent être trouvées aisément en photogrammétrie aérienne, situation où les écarts sur angles de rotation ne dépassent guère la dizaine de degrés, et où les distances et l'échelle sont connues à 10% près. C'est largement suffisant afin de lancer le processus itératifs de la linéarisation des équations. Ces techniques ont été largement décrites dans [Slama, 1980], [Wolf et Dewitt, 2000] et [Kraus et Waldaeusl, 1998].

A l'opposé de cette situation, en vision par ordinateur, en général les valeurs initiales ne sont pas connues et la contrainte de temps de calcul est très forte. Ceci écarte les méthodes itératives citées précédemment. L'article de [Fischler et Bolles, 1981] a introduit pour la première fois la résolution directe du relèvement dans la communauté de vision par ordinateur. D'autres variantes ont été ensuite publiées [Föstner, 1987], [DeMenthon et Davis, 1992]. Une synthèse complète ainsi qu'une analyse numérique approfondie de différentes méthodes de résolutions directes sont présentées par [Haralick *et al.*, 1994].

*Géométrie du relèvement*

La géométrie du relèvement est très simple. Comme on peut le voir sur la figure 2.15, le but est de calculer le sommet de prise de vue $O$. Pour cela nous disposons des 3 points terrain : $A$, $B$ et $C$, ainsi que de leurs projections sur l'image : $a$ $b$ et $c$.

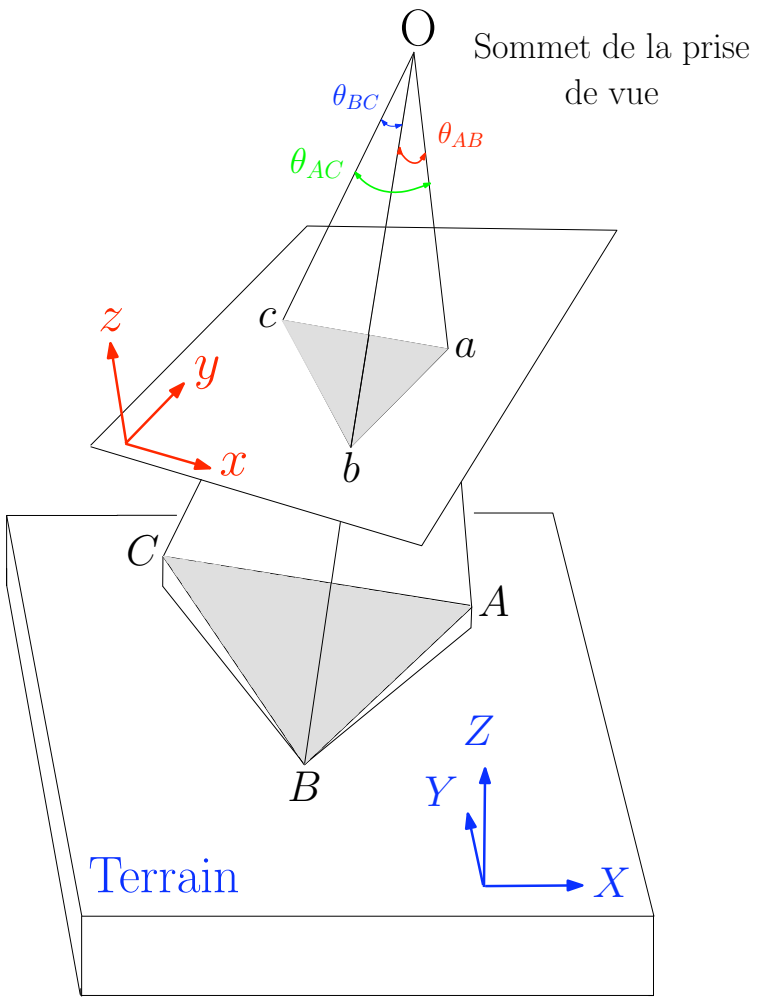



**Fig. 2.15.** Illustration de la géométrie du relèvement

On emploie ici le théorème d'Al-Kashi [Kennedy, 1947] [3]. Ce théorème est plus connu sous le nom de "loi des cosinus" qui généralise le théorème de Pythagore pour un triangle quelconque (équation 2.4) [4].

$$\begin{aligned} AB^2 &= OA^2 + OB^2 - 2OAOB\cos\theta_{AB} \\ AC^2 &= OA^2 + OC^2 - 2OAOC\cos\theta_{AC} \\ BC^2 &= OB^2 + OC^2 - 2OBOC\cos\theta_{BC} \end{aligned} \tag{2.4}$$

Dans ce système d'équations le but est de calculer la position de $O$.

La résolution du relèvement à partir de 3 points donne en général 4 positions possibles pour $O$. Cette ambigüité peut être enlevée en utilisant un quatrième point. C'est pour cela

3. Ghiyath ad-Din Jamshid Mas'ud al-Kashi, mathématicien et astronome perse (vers 1380, Kashan (Iran) - 1429, Samarcande (Ouzbékistan))

4. http://fr.wikipedia.org/wiki/Théorème_d'Al-Kashi

qu'il y a eu ensuite de nombreux auteurs [Triggs, 1999], [Ameller *et al.*, 2002], [Quan et Lan, 1999] qui ont recherché une résolution directe à partir de 4 ou même 5 points afin d'obtenir en fin du processus de calcul une solution unique.

Récemment [Nistér, 2004] propose un algorithme plus général pour la résolution du problème à l'aide de 3 points pour des caméras avec une perspective non centrale. Sa résolution inclut bien évidemment le problème classique des caméras à perspective centrale.

**Estimation de l'orientation relative, et compensation par faisceau de plusieurs vues, en vision par ordinateur**

Même si les topographes, géodésiens et photogrammètres utilisent la compensation par faisceaux depuis parfois plus de deux siècles, la communauté de vison par ordinateur a, depuis une vingtaine d'années, investi de nombreuses recherches dans ce domaine. La majorité des articles ont essayé d'optimiser les algorithmes de minimisations existants, on citera notamment [Tomasi et Kanade, 1992], [Martinec et Pajdla, 2007], [Ma *et al.*, 2003], [Hartley et Zisserman, 2004], [Kahl et Hartley, 2008]. Il existe par ailleurs dans cette communauté un logiciel puissant et très employé de compensation par faisceaux [Lourakis et Argyros, 2004]. Dans le cadre de cette thèse nous n'avons donc pas eu besoin d'approfondir ces domaines, car notre objectif principal est de fournir un positionnement et une orientation approchée pour un ensemble d'images. Une solution approchée robuste (car débarassée de toutes faute grossière) permet ensuite d'utiliser de façon nominale les algorithmes classiques d'aérotriangulation.

## 2.3 Extraction des primitives de l'image

L'orientation (relative et absolue) des images passe par l'identification et la mise en correspondance d'éléments structurants entre images et entre image et le terrain. Nous décrivons ici, de manière exhaustive, quelques techniques efficaces permettant :

- la détection de points d'intérêt dans les images et leurs mises en correspondance (paragraphe 2.3.1),

- l'extraction de segments dans les images (paragraphe 2.3.2).

### 2.3.1 Extraction de points d'intérêt

Une des avancées remarquables de ces dernières décennies dans le domaine du traitement des images a été la mise au point d'algorithmes de plus en performants pour l'extraction des points d'intérêt.

Il est évidemment de très important d'automatiser la détection des points d'intérêts dans le domaine de la photogrammétrie. En effet, pendant longtemps l'étape d'extraction de points d'intérêt (ou "points de liaison") se faisait entièrement à la main. Afin de pouvoir mettre en

place les images dans l'espace et de calculer leurs orientations et positions, il est primordial d'avoir en possession des points homologues sur les différentes images, c'est-à-dire des points qui correspondent à un même point du terrain.

Pour automatiser cette phase, on commence par extraire de chaque image des points faciles à pointer, et logiquement, si ces points ont été bien choisis, on parvient ensuite à mettre en correspondance la plupart des points d'une image avec ceux de l'autre dans la zone vue en stéréoscopie : on appelle ces points les points d'intérêt. Bien évidemment il est important d'éviter les points mal définis (p. ex. pris le long d'une bordure), ceux qui n'ont pas de définition géométrique stable (p. ex. bordure d'une ombre, intersection de lignes qui ne sont pas dans un même plan), etc. L'automatisation de l'extraction des points d'intérêt est directement liée à l'emploi d'images numériques, et l'un des premiers outils utilisés a été publié par Harris en 1986 ([Harris et Stephens, 1988]) : son détecteur est basé sur l'extraction automatique de coins, et si assez rapidement ses insuffisances ont été connues, sa simplicité d'implémentation en a fait un outil très employé. Néanmoins il a fallu attendre les résultats de deux décennies de recherche pour disposer d'une méthode réellement plus fiable, la méthode SIFT.

La méthode SIFT de [Lowe, 2004] (Scale Invariant Feature Transform) permet d'obtenir des points d'intérêt dont la détermination est très peu sensible à des changements, même importants, de facteurs d'échelle et d'orientation, et aussi assez peu sensibles aux variations locales de radiométrie (différences d'éclairement, différences de point de vue, etc.), toutes sortes de situations rencontrées très fréquemment en vision par ordinateur et où le détecteur de Harris est souvent tenu en échec. Elle est basée, entre autres, sur la considération suivante : dans une image, appliquer un changement d'échelle à un détail dont la répartition de densités radiométriques présente une forme gaussienne fournit un détail identique, à un facteur d'échelle radiométrique près.

Le traitement SIFT commence, pour chaque image, par une convolution successive avec des imagettes de gaussiennes de tailles progressivement croissantes. Puis une soustraction entre ces images convoluées successives est effectuée, permettant d'identifier un premier ensemble de points candidats, déjà a priori très peu sensibles aux changements d'orientation et d'échelle. Un premier filtrage est alors effectué pour éliminer les candidats ayant un contraste faible, ou qui sont mal localisables car situés sur un linéament.

L'étape suivante consiste, pour chacun de ces candidats, à identifier une ou plusieurs directions particulières au sein des gradients effectués dans toutes les directions autour du point. On construit alors, en utilisant ces directions comme références, un descripteur des gradients de l'environnement immédiat du point, sorte de carte d'identité du point en généralement 128 valeurs numériques, extrêmement peu sensible à l'orientation et à l'échelle du voisinage, ainsi qu'aux différences d'éclairement comme on le comprend aisément.

L'appariement consiste dans un premier temps à calculer toutes les distances euclidiennes à l'aide des 128 vecteurs entre tous les points de l'image 1 et tous les points de l'image 2. Ensuite pour chaque point de l'image 1, les deux points les plus proches (au sens du produit scalaire : points les plus comparables) sur l'image 2 sont extraits. Si le premier plus proche

voisin est au moins 6 fois plus près que le deuxième plus proche voisin, alors le premier voisin est retenu comme étant l'homologue du point sur l'image 1. Il est clair que ce processus est exponentiel en fonction du nombre de points de chaque image. C'est pour cela qu'il est indispensable d'utiliser des outils de type kd-tree, qui permettent d'augmenter de manière considérable la vitesse de détermination des plus proches voisins. Pour ce fait, il est possible d'utiliser la librairie C++ au nom de ANN [Arya *et al.*, 1998] qui est particulièrement optimisée dans ce but. Elle permet aussi la recherche dans un espace à plusieurs dimensions, en l'occurrence ici 128.

Les Figures 2.16, 2.17 illustrent l'extraction des points SIFT ainsi qu'un exemple d'appariement.

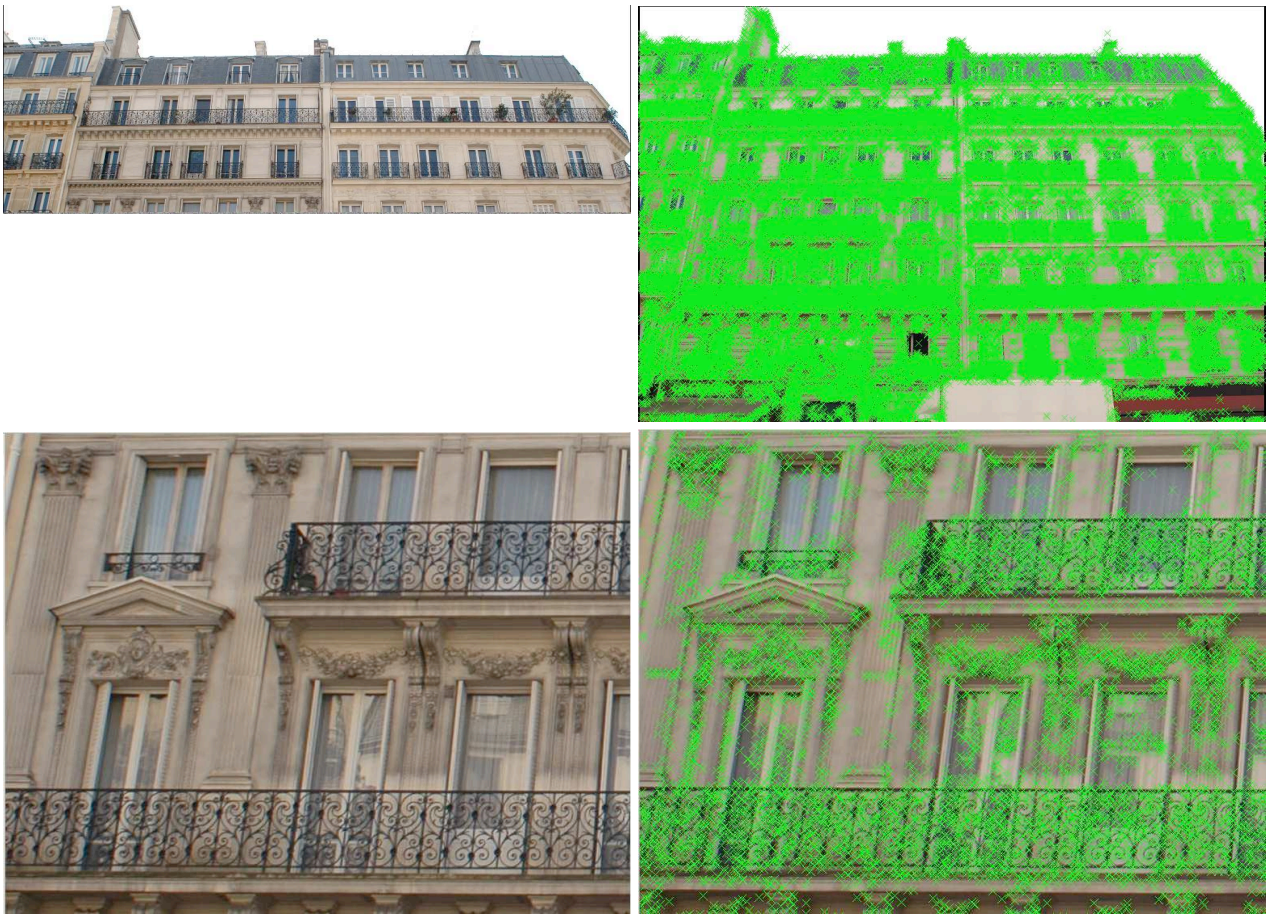


**Fig. 2.16.** Illustration d'extration des points d'intérêt avec la méthode SIFT. (Images fournies par Arnaud Le Bris)

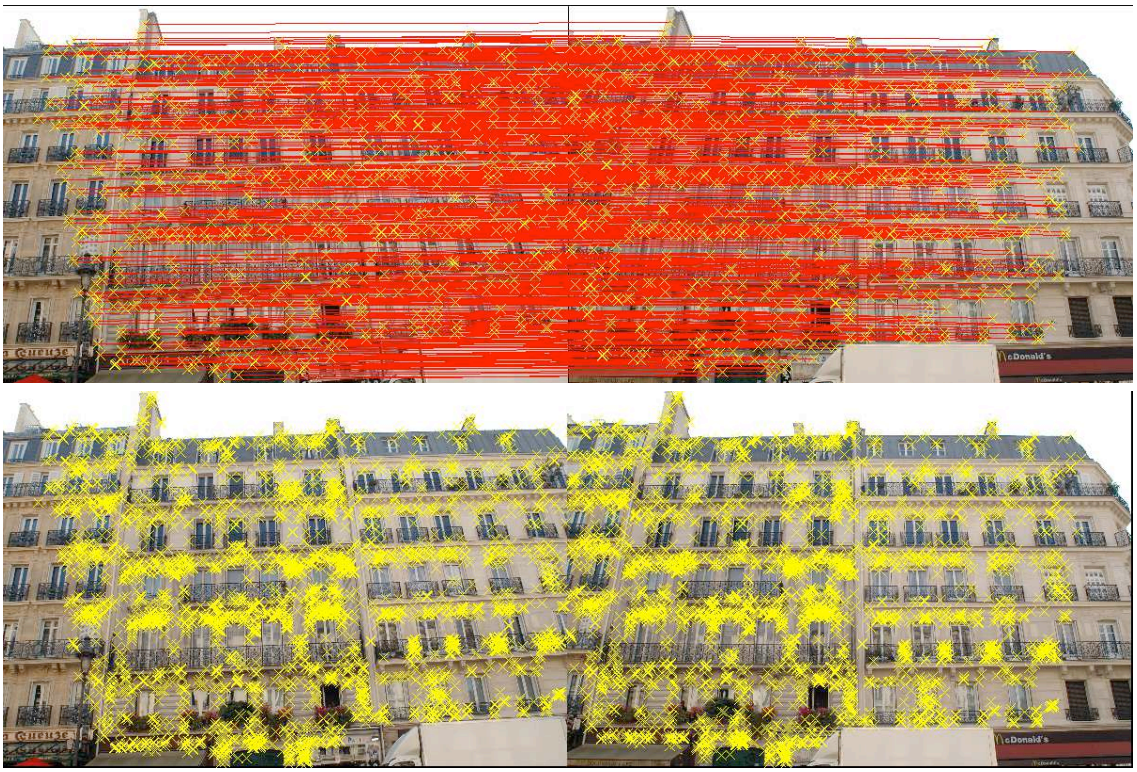


**Fig. 2.17.** Illustration d'appariement des points SIFT avec le kd-tree [Arya *et al.*, 1998]. (Crédit photo : Arnaud Le Bris)

Bien sur il existe d'autres algorithmes performants ([Mikolajczyk et Schmid, 2005], [Bay *et al.*, 2008]), mais dans notre travail nous avons essentiellement utilisé la méthode SIFT.

### 2.3.2 Extraction automatique des segments 2D de l'image

Les segments de l'image sont des primitives de l'image qui vont être très utilisées tout au long de cette étude. Comme on pourra le voir par la suite les segments seront très employés dans la recherche des points de fuite dans la partie II.

Il existe plusieurs algorithmes de détection des droites de l'image. Nous avons opté pour l'algorithme de [Deriche *et al.*, 1992]. Celui-ci permet, en plus de l'extraction des segments, le calcul des incertitudes des paramètres déterminants le segment. Nous précisons que nous nous plaçons en utilisateur de cet algorithme et qu'aucune recherche en quête de modification ou d'éventuelles améliorations n'a été menée.

Les principales étapes de détection des segments sont les suivantes :

1. Extraction de contours sur chaque image en utilisant un opérateur classique [Deriche, 1987]. Le paramètre $\alpha$ de Dériche est égal à 2.
2. Détection des maxima locaux dans la direction du gradient.
3. Seuillage à hystérésis à deux paramètres, seuil haut et seuil bas. Dans notre étude, ces seuils sont de 1 et 2.

4. Chaînage des pixels connexes
5. Polygonalisation : on utilise pour ce faire un processus de fusion itératif basé sur le résidu maximum de la régression orthogonale : on fusionne d'abord les polylignes dont la fusion fournit un résidu maximum minimal. Une tolérance sur la polygonisation permet d'arrêter l'algorithme lorsque la fusion envisagée fournit un résidu maximum supérieur à un seuil donné par l'utilisateur $s_pol$. Ici $s_pol$ est choisi à 0.5 pixel [Taillandier, 2004].
6. Estimation des paramètres des droites portant les segments en utilisant les résultats de [Deriche *et al.*, 1992].

*Estimation des paramètres de la droite et calcul de sa matrice variance-covariance*

Une régression est effectuée sur $n$ points $(x_i,\ y_i)$. $(\bar{x}, \bar{y})$ étant le barycentre de l'ensemble des $n$ points.

Une droite dans l'espace image peut être définie à l'aide des deux paramètres $\rho$ et $\theta$ (voir Figure 2.18).

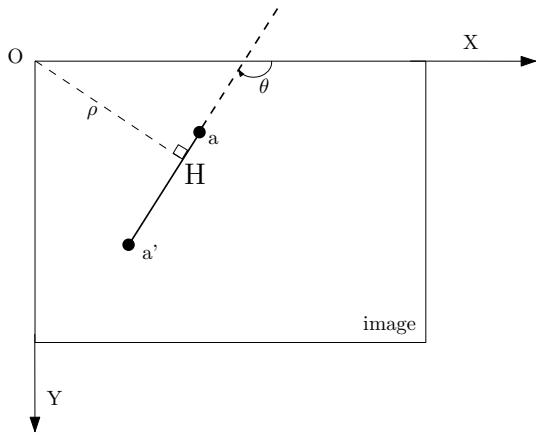



**Fig. 2.18.** Représentation d'une droite dans l'image.

Nous avons donc :

$$\begin{aligned} a &= \sum_{i=1}^{n}(x_i - \bar{x})^2, \\ b &= 2\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}), \\ c &= \sum_{i=1}^{n}(y_i - \bar{y})^2. \end{aligned} \tag{2.5}$$

On pourra donc calculer $\rho$ et $\theta$ :

$$\begin{aligned} \theta &= \frac{1}{2}\arctan(\frac{b}{a-c}), \\ \rho &= \bar{y}\cos(\theta) - \bar{x}\sin(\theta). \end{aligned} \tag{2.6}$$

La matrice de covariance des paramètres de la droite peut être estimée en supposant que les points détectés par l'opérateur de Canny-Deriche [Deriche, 1987] ont une variance donnée par :

$$\Lambda = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \tag{2.7}$$

On notera que le paramètre $\sigma$ peut être estimé à partir du rapport signal sur bruit de l'image et du paramètre du filtre de contours $\alpha$ : $\sigma = \frac{2}{\sqrt{\sqrt{\alpha}}}$.

Avec cette hypothèse, on obtient :

$$\Lambda_{\rho,\theta} = \frac{\sigma^2(a+c)}{(a-c)^2+b^2} \begin{bmatrix} 1 & -d \\ -d & d^2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{\sigma^2} \end{bmatrix} \tag{2.8}$$

Un des principaux résultats de cette étape est :
1. l'extraction de tous les segments de l'image
2. la matrice variance-covariance de chaque segment.

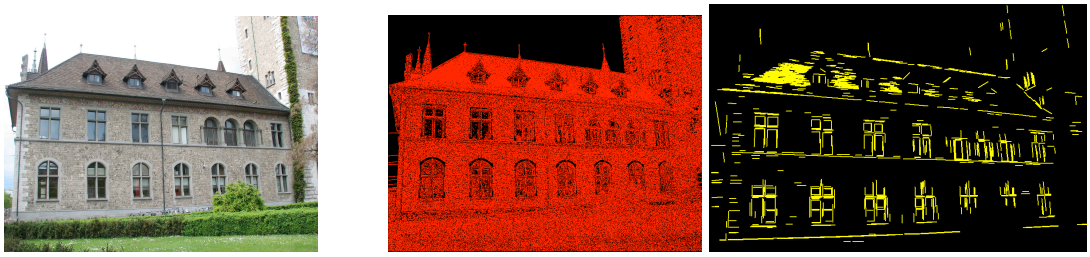


Image d'entrée
(Schweizerisches Landesmuseum, Zurich)

Extraction de contours

Détection de segments

**Fig. 2.19.** Illustration des différentes étapes d'extraction de segments

## 2.4 Conclusions

Dans ce chapitre, nous avons passé en revue l'historique de la photogrammétrie et de la vision par ordinateur. Puis nous avons vu les différentes approches pour résoudre une des étapes principales en photogrammétrie, c'est-à-dire l'orientation relative. Ensuite a été présenté un résumé de différentes méthodologies permettant de mettre un ensemble d'images dans la même référence, permettant de conclure qu'il existe à ce jour une quantité non négligeable de méthodes publiées différentes. Dans la suite, nous travaillerons à une amélioration de ces problèmes, en nous basant sur de nouveaux éléments de résolution.

# 3

# Introduction à la résolution des systèmes polynomiaux

## Sommaire



## 3.1 Introduction

Dans de nombreux domaines de la géomatique, par exemple en photogrammétrie, en topographie ou encore en géodésie, les systèmes d'équations rencontrés ne sont pas linéaires. Il n'y a d'ailleurs que très peu d'applications dans ces domaines qui reposent sur des équations linéaires. Une des méthodes les plus connues et employées pour résoudre les systèmes polynomiaux est la résolution analytique qui exploite une fonction $f$ ainsi que ses dérivées, p. ex. la méthode de Newton qui est itérative, ou encore la minimisation locale de $f$. L'inconvénient majeur de ces méthodes est dû au fait qu'elles nécessitent des valeurs initiales approchées, parfois très difficiles à obtenir. Par ailleurs il est important de rappeler que pendant longtemps, faute de disposer des moyens de calcul actuels, les scientifiques n'avaient pas d'autre choix que de linéariser leurs équations, et de les résoudre sous cette formalisation ayant une solution facile à trouver.

Une autre façon de trouver les solutions d'un système polynomial est d'avoir recours à une résolution algébrique. Les bases de Gröbner que nous présentons ici rentrent dans cette catégorie, depuis quelques dizaines d'années elles ont bénéficié d'avancées remarquables grâce à l'augmentation de la puissance de calcul. Et aujourd'hui grâce à de tels outils mathématiques, il est tout à fait possible de résoudre des équations non-linéaires de manière directe.

Dans ce chapitre, nous effectuons un bref rappel des bases de Gröbner, encore peu connues, mais dont l'importance est considérable. Cette méthode permet en effet de re-évaluer l'ensemble des méthodes classiques de beaucoup de secteurs techniques en repartant de zéro, et il importe de bien la comprendre puisqu'une partie de nos travaux est basée dessus. Ces éléments mathématiques ont été approfondis avec le concours de Amir Hashemi.

## 3.2 Résolution de systèmes polynomiaux

La résolution des systèmes polynomiaux est un problème très important qui se retrouve fréquemment dans plusieurs domaines des sciences fondamentales et d'ingénieur. La plupart des travaux dans ce domaine sont consacrés à trois approches : *les bases de Gröbner, les systèmes triangulaires et la décomposition algébrique cylindrique*, cf. [Cox *et al.*, 2004] et [Cox *et al.*, 2007b]. Dans ce chapitre, nous rappelons brièvement une première approche pour le lecteur non familier de ce sujet.

Soit un système d'équations polynomiales donné :

$$f_1(x_1, \ldots, x_n) = \cdots = f_k(x_1, \ldots, x_n) = 0$$

où $f_1, \ldots, f_k \in K[x_1, \ldots, x_n]$ sont quelques polynômes, et $K$ est un corps arbitraire.

La *la résolution* de ce système couvre les questions suivantes :

1. Le système $f$ a-t-il des solutions ?
2. Si oui, sont elle en nombre fini ou infini ?
3. Si elle sont en nombre fini, comment les calculer ?

En d'autres termes, *la résolution* de ce système veut dire, en gros, trouver une description de la variété

$$\mathrm{V}(F) = \{(a_1, \ldots, a_n) \in K^n \mid f_1(a_1, \ldots, a_n) = \cdots = f_k(a_1, \ldots, a_n) = 0\}$$

où $F = f_1, \ldots, f_k$. Résoudre de façon purement numérique signifie trouver pour chaque point de $\mathrm{V}(F)$ une approximation en virgule flottante pour une precison donnée. Si $\mathrm{V}(F)$ est fini, alors le but pour un prétraitement algébrique devrait être de transformer le système initial $F$ en un système dont la variété couvre $\mathrm{V}(F)$, mais qui est soluble numériquement et si possible avec des équations plus simples.

La variété $\mathrm{V}(F)$ *ne dépend pas* de $F$, mais de l'idéal engendré par $F$. Rappelons ici que l'idéal produit par un ensemble fini de polynômes $f_1, \ldots, f_k$ est défini ainsi :

$$\langle f_1, \ldots, f_k \rangle = \left\{ \sum_{i=1}^{k} h_i f_i \mid h_i \in K[x_1, \ldots, x_n] \right\}. \tag{3.1}$$

Donc, on peut prendre, au lieu de $F$, un autre ensemble générateur, appelé *base de Gröbner*. Cette recherche a été menée par Buchberger en 1970, cf. [Buchberger, 1965]. Dans la suite, nous rappelons en premier les définitions et les propriétés des bases de Gröbner (paragraphe

3.3). Ensuite, nous rappelons la définition et les propriétés des systèmes zéro-dimensionnels (paragraphe 3.4). Finalement, nous présentons brièvement un solveur algébrique qui utilise les bases de Gröbner pour trouver les racines réelles d'un système zéro-dimensionnel (paragraphe 3.5).

## 3.3 Bases de Gröbner

La notion de base de Gröbner a été introduite par Buchberger en 1965, qui a fourni le premier algorithme de calcul, cf. [Buchberger, 1965]. Soit $R = K[x_1, \ldots, x_n]$ un anneau de polynômes où $K$ est un corps arbitraire. Soit $f_1, \ldots, f_k \in R$ une séquence de $k$ polynômes et $I = \langle f_1, \ldots, f_k \rangle$ un idéal de $R$ engendré par les $f_i$. Nous avons besoin aussi d'un ordre de rangement des monômes sur $R$. Rappelons ici la définition des deux principaux types de rangement de monômes. Le *rangement lexicographique* (LEX), noté $\prec_{LEX}$, est utilisé pour résoudre des systèmes polynomiaux et le *rangement lexicographique inverse* (DRL), noté $\prec_{DRL}$, qui est un rangement spécial des monômes ayant des propriétés de calcul intéressantes, cf. [Cox *et al.*, 2007b] pour la définition d'autres rangements. Nous dénotons respectivement par $\deg(m)$(resp. $\deg_i(m)$) le degré total (resp. le degré dans $x_i$) d'un monôme $m$. Si $m$ et $m'$ sont des monômes, alors,

- $m \prec_{LEX} m'$ si et seulement si le premier élément non nul de la séquence $(\deg_1(m') - \deg_1(m), \ldots, \deg_n(m') - \deg_n(m))$ est negatif,
- $m \prec_{DRL} m'$ si et seulement si le dernier élément non nul de la séquence $(\deg_1(m') - \deg_1(m), \ldots, \deg_n(m') - \deg_n(m), \deg(m) - \deg(m'))$ est negatif.

Dans la suite, soit $\prec$ un rangement de monômes. Soit $in(f) \in R$ le premier monôme (de plus haut degré) d'un polynôme $f \in R$ selon $\prec$ et $in(I) = \langle in(f) \mid f \in I \rangle$ l'ideal initial de $I$.

**Définition 3.1 (Base de Gröbner).** *Un sous-ensemble $G \subset I$ est une base de Gröbner de $I$ selon $\prec$ si $\langle in(G) \rangle = in(I)$.*

**Définition 3.2 (Base de Gröbner réduite).** *Une base de Gröbner $G$ est dite réduite si pour tout $g \in G$, $g$ est monique et qu'aucun monôme $g$ n'est dans $\langle in(G \setminus \{g\}) \rangle$.*

**Proposition 3.3.** *Tout idéal a une base Gröbner réduite unique.*

*Preuve.* Cf. [Cox *et al.*, 2007b], Proposition 6, page 92.

Il faut noter que le concept de base de Gröbner généralise trois techniques classiques : l'élimination de Gauss pour résoudre des systèmes linéaires d'équations, l'algorithme Euclidien pour calculer le plus grand diviseur commun de deux polynômes à une seule variable, et l'algorithme du simplex pour la programmation linéaire.

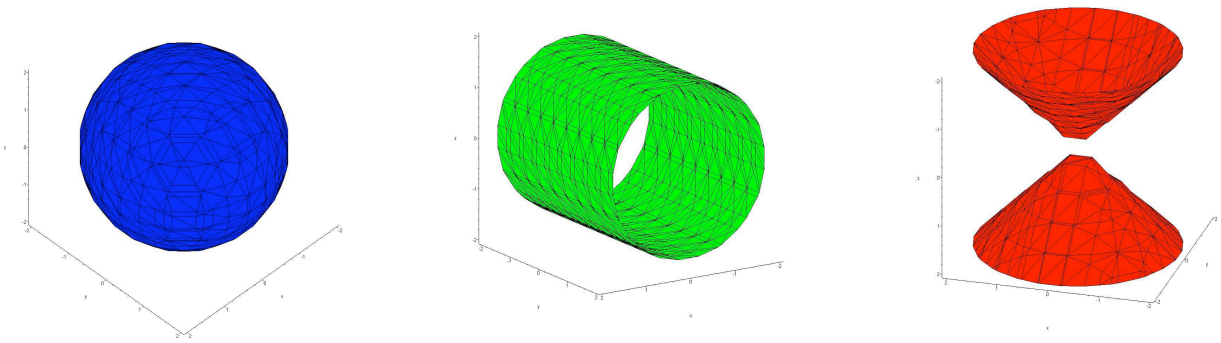
Il y a plusieurs algorithmes possibles pour calculer efficacement les bases de Gröbner. Le plus traditionnel est l'algorithme de Buchberger, il a plusieurs variantes et il est utilisé dans la plupart des outils informatiques de calcul algèbrique généraux comme Maple[1], Mathematica[2], Singular [Greuel *et al.*, 2005], Macaulay2 [Grayson et Stillman, 1996],

1. Maple : www.maplesoft.com
2. Mathematica : http://www.wolfram.com/products/mathematica/index.html

Cocoa [3] et le logiciel Salsa [4]. Ici nous utilisons le logiciel Salsa avec l'algorithme F4 [Faugère, 1999]. L'algorithme F4 de Faugère est basé sur l'usage intensif de méthodes d'algèbre linéaire.

*Exemple*

Avant de continuer, pour une meilleure compréhension, nous illustrons ces définitions de géométrie algébrique avec un exemple par ailleurs très simple et facile à résoudre. Supposons que nous recherchions l'intersection d'une sphère, d'un cylindre et d'un cône de révolution centrés sur le même point (Figure 3.1).



Sphère Cylindre Cône de révolution

**Fig. 3.1.** Représentation des trois figures géométriques utilisées dans notre exemple.

Nous cherchons l'intersection de ces trois ensembles (Figure 3.2), et on montre facilement qu'elle est composée de 8 points.

3. Cocoa, A System for doing Computations in Commutative Algebra : http://cocoa.dima.unige.it
4. Salsa, Solvers for ALgebraic Systems and Applications : http://fgbrs.lip6.fr/salsa/

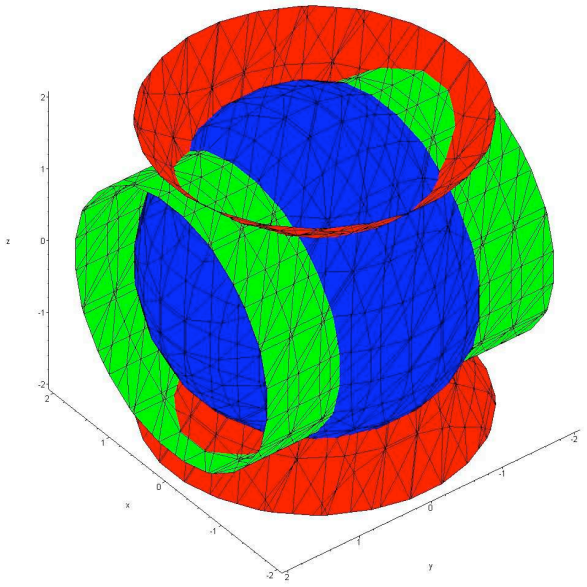


**Fig. 3.2.** Illustration de l'intersection des éléments géométriques de la Figure 3.1.

La sphère est définie par l'équation : $x^2 + y^2 + z^2 = 4$, le cylindre par $z^2 + x^2 = 3$ et le cône de révolution par $x^2 + y^2 = z^2$.

Donc nous avons le système d'équations polynomiales suivant :

$$\begin{aligned} f_1 &= x^2 + y^2 + z^2 - 4 = 0 \\ f_2 &= z^2 + x^2 - 3 = 0 \\ f_3 &= x^2 + y^2 - z^2 = 0, \end{aligned} \tag{3.2}$$

avec comme inconnues $(x, y, z)$. L'ordre choisi pour les monômes est l'ordre $DRL$, donc $z \prec y \prec x$. Les bases de Gröbner sont calculées pour ce système et nous obtenons :

$$GB_1 = z^2 - 2, GB_2 = -1 + y^2, GB_3 = x^2 - 1. \tag{3.3}$$

Résoudre le système d'équations 3.2 reviendra à résoudre le système d'équations 3.3. Dans la suite de ce chapitre nous reviendrons à cet exemple, en appliquant au fur et à mesure les nouvelles définitions.

## 3.4 Systèmes de dimension zéro

Nous rappelons maintenant la définition et les propriétés d'un système de dimension zéro. Ce cas spécifique est fondamental pour beaucoup d'applications en ingénierie.

**Définition 3.4 (Système de dimension zéro).**
*Avec les notations du paragraphe précédent, un système* $\{f_1 = \cdots = f_k = 0\}$ *est dit de dimension zéro s'il a un nombre fini de solutions dans* $K^n$.

Cette définition est équivalente à la propriété suivante : pour chaque $1 \leq i \leq n$, il y a un $m_i \geq 0$ tel que $x_i^{m_{iIn(g)|}}$ pour quelques $g$ dans toute base de Gröbner $I$. Cette équivalence est appelée *critère de finitude*, qui fournit un critère algorithmique pour déterminer quand un système donné est de dimension zéro (cf. [Cox *et al.*, 2007b], page 234).

Soit dans notre exemple le système d'équations polynomiales 3.2 : $f_1,\ f_2,\ f_3 \in R = \mathbb{C}[x, y, z]$, où $\mathbb{C}$ est le corps de nombres complexes, et soit aussi $I$ l'idéal engendré par ces polynômes. Grâce à la propriété que la division par l'idéal $I$ est bien définie quand nous le faisons par rapport à une base de Gröbner de $I$, nous pouvons considérer l'espace de tous les restes des division par $I$ (cf. [Cox *et al.*, 2007b]). Cet espace est appelé l'*anneau quotient* de $I$, et nous le notons $A = R/I$. Il est bien connu que si $I$ est racine, alors le système $f_1 = f_2 = f_3 = 0$ a un nombre fini de solutions $N$ si la dimension de $A$ comme un $\mathbb{C}$-espace vectoriel est $N$ (cf. [Cox *et al.*, 2007b], proposition 8 page 235). Nous pouvons vérifier facilement par la fonction `IsRadical` de MAPLE que $I$ est racine. Une base pour $A$ comme espace vectoriel est obtenue à partir de $in(I)$ par ([Cox *et al.*, 2007b], théorème 6, page 234).

Remarquons que tout ordre de rangement peut être choisi pour calculer une base de Gröbner de $I$ et ainsi obtenir $in(I)$. Les éléments de $B$ sont appelés *base de monômes* ou *monômes standard* de $I$ par rapport au classement choisi des monômes. Par exemple, si nous choisissons $\prec_{DRL}$ alors l'ensemble :

$$B = [1, z, y, x, yz, xz, xy, xyz] \tag{3.4}$$

est une base pour $A$ en tant que $\mathbb{C}$-espace vectoriel.

Par conséquent, nous pouvons conclure que le système $f_1 = f_2 = f_3 = 0$ a 8 solutions.

Nous rappelons brièvement ici la méthode des valeurs propres pour résoudre le système $f_1 = f_2 = f_3 = 0$, cf. [Cox *et al.*, 2004] page 56, pour plus de détails. Pour tout $f \in R$, nous notons $[f]$ le coset de $f$ dans $A$. Nous définissons $m_f : A \longrightarrow A$ par la règle suivante :

$$m_f([g]) = [f].[g] = [fg] \in A$$

Comme l'idéal produit par les $f_i$ est de dimension zéro, alors $A$ est un $\mathbb{C}$-espace vectoriel de dimension finie, et nous pouvons représenter $m_f$ par une matrice qui est appelée la *matrice d'action* de $f$.

**Théorème 3.5 (Stickelberger).** *Les valeurs propres de* $m_f$ *sont exactement les* $f(\alpha)$ *où* $\alpha \in \mathrm{V}(I)$.

*Preuve.* Cf. [Cox *et al.*, 2004], Theorème 4.5, page 59.

Conformément à ce théorème, la $i$-ième coordonnée de tout $\alpha \in \mathrm{V}(I)$ peut être déduite des valeurs propres $m_{x_i}$, mais la question de trouver toutes les coordonnées de tous le $\alpha \in \mathrm{V}(I)$ à partir des valeurs propres de $m_{x_1}, \ldots, m_{x_n}$ n'est pas explicite et difficile à prouver.

## 3.5 Représentation Univariée Rationnelle (RUR)

Une fois la base de Gröbner d'un système zéro-dimensionnel calculée, différentes solutions existent pour trouver les racines du système. Par exemple, la méthode qui résout le système avec l'aide de bases de Gröbner selon $\prec_{LEX}$. Une autre façon très courante de résoudre des systèmes polynomiaux utilise les vecteurs propres et les valeurs propres (cf. paragraphe précédent). Ici l'accent est mis sur une autre méthode, appelée *Représentation Univariée Rationnelle* (en abrégé RUR). Cette représentation est la solution la plus simple pour représenter symboliquement les racines d'un système zéro-dimensionnel sans perdre d'information (racines réelles ou multiples) puisqu'on peut obtenir toute l'information sur les racines du système en résolvant des polynômes à une seule variable. La RUR a été introduite en premier par Leopold Kronecker [Kronecker, 1931], mais n'a commencé que récemment à être utilisée en calcul algèbrique informatique [Gonzalez-Vega, 1997], [Rouillier, 1999].

Considérant un système de dimension zéro $I = \langle f_1, \ldots, f_k \rangle$ où $f_i \in \mathbb{Q}[x_1, \ldots, x_n]$ et $\mathbb{Q}$ est le corps des nombres rationnels, une RUR de $\mathrm{V}(I)$ prend la forme suivante :

$$f_t(T) = 0, x_1 = \frac{g_{t,x_1}(T)}{g_{t,1}(T)}, \ldots, x_n = \frac{g_{t,x_n}(T)}{g_{t,n}(T)}$$

où $f_t, g_{t,1}, g_{t,x_1}, \ldots, g_{t,x_n} \in \mathbb{Q}[T]$ ($T$ est une nouvelle variable). Elle est définie de façon unique par rapport à un polynôme donné t qui sépare $\mathrm{V}(I)$ (injectif sur $\mathrm{V}(I)$), le polynôme $f_t$ étant nécessairement le polynôme caractéristique de $m_t$ (voir paragraphe précédent) dans $\mathbb{Q}[x_1, \ldots, x_n]/I$ [Rouillier, 1999].

Pour calculer une RUR, il y a deux problèmes à résoudre :
- trouver un élément de séparation $t$
- étant donné un polynôme quelconque $t$, calculer une RUR candidate $f_t, g_{t,1}, g_{t,x_1}, \ldots, g_{t,x_n}$ donc si $t$ est un polynôme de séparation, alors la candidate RUR est une RUR.

Le but est de calculer toutes les racines réelles du système (et seulement ces racines), en donnant une approximation numérique d'une précision arbitraire (choisie par l'utilisateur) des coordonnées. Beaucoup d'algorithmes efficaces ont été développés pour permettre les calculs de type RUR. Plus de détails peuvent être trouvés facilement dans la littérature, sachant qu'une explication complète peut être trouvée dans [Ouchi *et al.*, 2003],[Rouillier, 1999]. Une mise en oeuvre de l'algorithme de Rouillier pour le calcul de RUR peut être trouvée dans le logiciel SALSA [5].

5. Salsa, Solvers for ALgebraic Systems and Applications :http://fgbrs.lip6.fr/salsa/

En revenant à notre exemple et en utilisant la librairie Salsa pour le calcul du RUR, nous obtenons les racines réeles du système d'équations 3.2. Le polynôme à une seule variable ($P$) est le suivant :

$$P^8 - 148P^6 + 6902P^4 - 113716P^2 + 508369 = 0. \tag{3.5}$$

Une fois l'équation 3.5 résolue, nous pourrons alors trouver les solutions pour $x$, $y$ et $z$ :

$$\begin{aligned} x &= \frac{P^6 - 39P^4 + 171P^2 + 24955}{P^7 - 111P^5 + 3451P^3 - 28429P}, \\ y &= \frac{2P^6 - 90P^4 - 1842P^2 + 41354}{P^7 - 111P^5 + 3451P^3 - 28429P}, \\ z &= \frac{8P^6 - 808P^4 + 22200P^2 - 154008}{P^7 - 111 * P^5 + 3451P^3 - 28429P}. \end{aligned} \tag{3.6}$$

Enfin, comme on l'a vu précédemment notre système a 8 solutions qui sont les suivantes :

1. $y = -1, x = -1, z = -\sqrt{2}$
2. $y = -1, z = -\sqrt{2}, x = 1$
3. $y = 1, x = -1, z = -\sqrt{2}$
4. $y = 1, z = -\sqrt{2}, x = 1$
5. $y = -1, x = -1, z = \sqrt{2}$
6. $y = -1, x = 1, z = \sqrt{2}$
7. $y = 1, x = -1, z = \sqrt{2}$
8. $y = 1, x = 1, z = \sqrt{2}$

Note : Aujourd'hui la librairie Salsa a été intégrée dans la nouvelle version de Maple 12.

## 3.6 Construction manuelle des Bases de Gröbner avec l'aide de la Matrice de Macaulay

Précédemment nous avons vu ce qu'était une base de Gröbner, et aussi qu'il existait des logiciels mathématiques pour l'obtenir.

Les algorithmes de résolutions polynomiaux dans ces librairies mathématiques sont mis en oeuvre pour pouvoir résoudre tous les systèmes d'équations en général. Or les systèmes que nous sommes amenés à traiter sont toujours les mêmes. Par exemple dans le cas de l'orientation relative à l'aide de la contrainte de coplanarité, seules les coordonnées des points homologues changent. Le système d'équations, lui, reste le même.

C'est pour cela que nous nous sommes intéressés à la construction des bases de Gröbner pour un système spécifique donné sans passer par une librairie existante. Ceci nous conduit à préciser l'emploi de la matrice de Macaulay, et nous expliquons comment nous pourrions l'utiliser pour calculer la base de Gröbner d'un idéal. Avec les notations de la section précédente, nous considérons l'idéal $I$ engendré par les $f_i$, et $\prec$, l'ordre DRL sur les monômes.

Nous supposons que nous connaissons le degré maximum $d$ des monômes qui apparaissent dans la représentation des éléments de la base de Gröbner de $I$ en les $f_i$. Notons que ce degré est le degré maximal des monômes qui paraissent dans le calcul de la base de Gröbner de $I$. Nous pouvons construire la matrice de *Macaulay* $M_d(f_1, \ldots, f_k)$ (pour faire court nous la dénotons par $M_d$) comme suit : écrivons horizontalement tous les monômes de degré au plus égal à $d$, rangés selon $\prec$ (le premier est le plus grand). Ainsi, chaque colonne de la matrice est indexée par un monôme de degré au plus $d$. Multiplions chaque $f_i$ de 1 à $k$ par n'importe quel monôme $m$ de degré au plus $d - \deg(f_i)$, et écrivons les coefficients de $mf_i$ sous leurs monômes correspondants, ce qui donne donc une ligne de la matrice. Les lignes sont rangées : la rangée $mf_i$ est avant $uf_j$ si ou bien $i < j$, ou $i = j$ et $m \prec u$.

Pour toute ligne dans la matrice, considérons l'indexation du monôme de la première colonne non nulle de cette ligne. Il est appelé le monôme de tête de la ligne, et c'est le monôme de tête du polynôme correspondant.

$$M_d = \begin{array}{c} \\ \vdots \\ mf_i \\ \vdots \end{array} \overset{\text{les monômes de degré au plus } d}{\begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}}$$

*L'élimination de Gauss* appliquée à cette matrice conduit à une base de Gröbner de $I$ (cf. Lazard [1983]). En effet, appelons $\tilde{M}_d$ la forme d'élimination de Gauss de $M_d$, où la seule opération élémentaire possible dans une ligne est l'addition d'une combinaison linéaire des lignes antérieures. Maintenant, considérons tous les polynômes qui correspondent à une ligne dont le terme de tête n'est pas le même dans $M_d$ et dans $\tilde{M}_d$, alors l'ensemble de ces polynômes est une base de Gröbner de $I$.

L'utilisation de la matrice de Macaulay dans la construction d'un solveur algébrique spécifique, sera mise en œuvre dans les chapitres 8 et 9.

## 3.7 Conclusion

Dans ce chapitre nous avons passé en revue les méthodes de résolution des systèmes polynomiaux. Aujourd'hui nous pouvons affirmer que toutes les équations non-linéaires employées en photogrammétrie peuvent être résolues sans l'intermédiaire d'une linéarisation. Les outils logiciels mathématiques existants permettent la résolution de ces systèmes, et ceci sans pour autant être spécialiste en mathématiques. Nous verrons plus loin la mise en œuvre de ces concepts dans la résolution directe de l'orientation relative.

# Partie II

# Estimation de l'orientation de la caméra à partir d'une seule image

# 4

# Détection automatique des points de fuite : Une aide à l'estimation de l'orientation

## Sommaire



## 4.1 Introduction

Le but recherché dans cette partie est de pouvoir extraire des informations concernant la prise de vue, en n'utilisant qu'une seule image. Nous reformulons de la sorte : quelle information sur la géométrie de la prise de vue peut-on tirer d'une seule image ? Une réponse à cette question est : l'orientation de la prise de vue, à l'aide de la localisation des points de fuite, qui sont des primitives de haut niveau.

Dans la géométrie conique caractéristique de la vision humaine ou de la photographie, les lignes parallèles de l'espace objet (aussi appelé l'espace terrain) se traduisent dans chaque image par des faisceaux de droites qui concourent sur des points de fuite. A chaque point de fuite une direction 3D est associée.

Les segments parallèles, qui se traduisent dans les images par des points de fuite, sont caractéristiques des objets fabriqués par l'homme, tout particulièrement les bâtiments, dans

lesquels la quasi totalité des lignes visibles correspondent à des éléments strictement horizontaux ou verticaux, très clairement liées à la géométrie adoptée pour gérer la présence de la gravité dans son environnement.

La connaissance combinée de certaines de ces primitives, en l'occurrence du point de fuite correspondant à la direction verticale et des points de fuites pour des directions horizontales, permet d'extraire un ensemble d'orientations de plans par rapport au repère caméra. Ceci réduit de manière considérable la complexité de résolution des paramètres extrinsèques de la caméra pour un couple d'image stéréoscopiques. Car pour chaque couple nous aurons d'autant plus d'informations sur leur rotation relative que nous aurons de points de fuite, localisés dans l'image, puis correctement identifiés.

Et c'est justement cette information que nous cherchons à obtenir dans notre quête d'estimation d'éléments d'orientation relative, dont la stratégie globale développée vise à réduire la complexité, en tirant profit de ces points de fuite.

Au total : les points de fuite permettent d'une part d'estimer la rotation relative du cliché par rapport à la scène, et d'autre part d'extraire des primitives robustes (segments horizontaux, segments verticaux etc.) de bas niveau qui peuvent ensuite être regroupées pour construire des objets plus complexes comme des façades (fenêtres, étages, etc.) dans le cas d'imagerie urbaine. La classification et le tri des différentes familles de segments peut être utilisée dans les algorithmes de mise en correspondance de segments.

La présente partie permettra d'introduire le travail développé pour extraire de façon automatique les points de fuite dans des images selon deux méthodes nouvelles, présentées dans les chapitres 5 et 6. Mais tout d'abord, la littérature sur l'extraction de points de fuites dans les images est très riche, et dans la section 4.3, un résumé des méthodes existantes est présenté. Auparavant, dans le paragraphe 4.2 nous retracerons les travaux publiés au cours des derniers siècles. Enfin pour finir nous donnerons un panorama d'applications des points de fuite dans le paragraphe 4.4.

## 4.2 Un peu d'histoire

L'utilisation du point de fuite pour définir l'orientation du sommet de prise de vue n'est pas quelque chose de récent. Sa première utilisation dans ce sens date du XV$^{ème}$ siècle. Ainsi Leon Battista Alberti (1404 -1472) écrivit un traité [Batista, 1452] qui repose sur l'expérience de Brunelleschi (1415), qui a peint le baptistère de Florence, selon des règles précises. Ensuite il a convié la foule à venir comparer, au travers d'un trou pratiqué dans le tableau et par l'intermédiaire d'un miroir, le tableau et le baptistère.

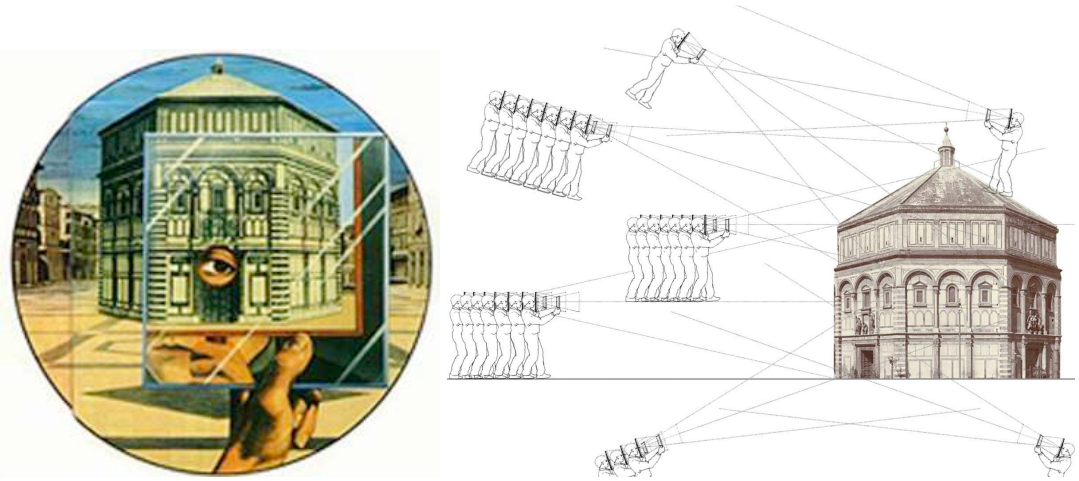


**Fig. 4.1.** L'expérience de Brunelleschi

Cette expérience révèle les principes de cet art, celui de la "perspective centrale", elle pose l'existence d'un centre, un "point de vue" unique où il faut se rendre pour voir ce que le peintre a vu, l'œil doit se placer face à un point précis de la peinture, "le point central" du tableau [Philippe, 1992], ce que dans un langage photogrammétrique nous appelons "sommet de prise de vue".

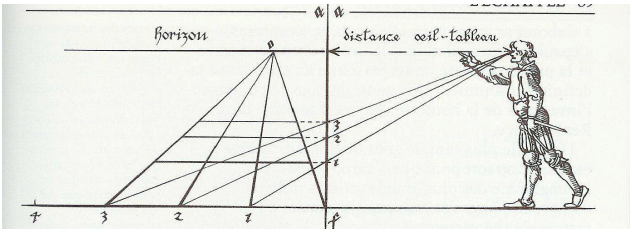



**Fig. 4.2.** La construction d'Alberti [Philippe, 1992]

L'outil de mise en perspective, proprement dit, vient de l'Allemand Albrecht Dürer [Benallal, 2002] qui publia un ouvrage agrémenté de planche assez explicites décrivant sa technique de mise en projection perspective.

Ce que ces illustres géomètres ont voulu déterminer était donc l'emplacement de l'œil du peintre. Dans un contexte plus moderne, les travaux issus de la photogrammétrie et de la communauté de vision par ordinateur cherchent aussi à déterminer en fonction des images (l'analogue du tableau d'autrefois) la position de la caméra (analogue de l'œil) au moment de la prise de vue (peinture). C'est ce qui fait l'objet de notre étude.

## 4.3 Etat de l'art de la détection des points de fuite

Les différents algorithmes de détection de points de fuite peuvent être différenciés les uns avec les autres en fonction de leurs espaces de travail, permettant donc de procéder à une

classification des méthodes. Il faut noter que la littérature dans ce domaine est très riche, de nombreux algorithmes ont été proposés.

L'évolution des techniques de détection des points de fuite, au cours des années, est dû à notre avis à deux facteurs. Tout d'abord, les avancées dans le domaine du traitement de l'image : la détection de segments peut se faire aujourd'hui de manière rapide et précise permettant ainsi l'utilisation d'images à haute résolution, ainsi que la vidéo. Le second facteur est l'utilisation de méthodes d'estimation robuste, de type RanSac [Fischler et Bolles, 1981] (voir Annexe A). Au départ les techniques de détection utilisaient des espaces d'accumulation afin de pouvoir extraire ces points de fuite. Mais petit à petit, avec l'utilisation de plus en plus fréquente du RanSac, des algorithmes bien plus rapides et moins gourmands en mémoire on vu le jour. La méthode RANSAC (en anglais Random Sample Consensus) est une méthode de vote probabiliste qui a été proposée afin de réduire le temps de calcul des méthodes de votes classiques (comme par exemple la transformée de Hough).

Les méthodologies employées couramment travaillent principalement dans différents espaces, ici nous citerons les trois espaces les plus rencontrés dans la littérature.

1. Ceux qui travaillent dans l'espace de la sphère de Gauss.
2. Ceux qui travaillent dans l'espace de la transformée de Hough.
3. Ceux qui travaillent directement dans l'espace image.

*Travail dans l'espace de la sphère de Gauss*

C'est l'espace le plus employé dans la littérature. Introduite en 1983 par [Barnard, 1983], la sphère de Gauss permet de travailler dans un espace fini, y compris pour les images de points eux-mêmes à l'infini. Chaque segment de l'image (voir Figure 4.3) est modélisé par la normale au plan formée par les deux extrémités du segment et le centre optique de la caméra. Par ailleurs une sphère unité est formée autour du centre optique de la caméra. Les plans formés à partir de chaque segment coupent cette sphère selon des grands cercles.

L'idée de Barnard est de dire que si un ensemble de segments représentant des lignes parallèles du monde réel se coupent sur le point dans l'image, alors les grands cercles correspondants sur la sphère se couperont au même endroit. Si on trace un vecteur qui part du centre optique et qui passe par ce lieu d'intersection des cercles sur la sphère, ce vecteur perce le plan image sur le point de fuite.

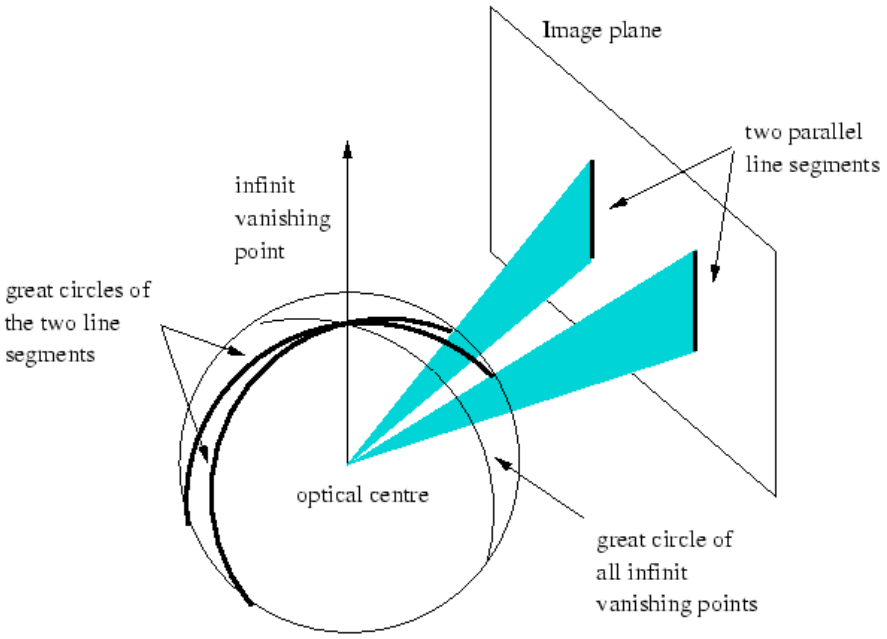



**Fig. 4.3.** Illustration de la sphère de Gauss d'après [Shufelt, 1999]

Comme les méthodes de détection de segments de l'image ne fonctionnent pas de manière 100 % fiable, les intersections des grands cercles sur la sphère ne donnent pas lieu à un point unique. C'est alors que les méthodes d'accumulations interviennent. Comme leurs noms l'indiquent, ces méthodes créent des espaces d'accumulation discrétisés sur la sphère et détectent les cellules où il y a le plus de grands cercles qui passent. On peut très bien imaginer la lourdeur de ce type de procédé, en plus de la complexité mathématique d'un travail sur une sphère discrétisée. De plus la définition de la taille des cellules d'accumulation est difficile et doit s'adapter à tous les types d'images.

L'idée de la sphère de Gauss étant très astucieuse, nombre d'auteurs au fil des années ont essayé de donner des améliorations en temps et en réduction de complexité de calcul. Parmi ces auteurs on peut citer [Magee et Aggarwal, 1984] qui accumulent en projection deux par deux l'intersection des segments sur la sphère de Gauss. Cette méthode est très gourmande en termes de temps de calcul, mais considérée comme très précise.

Plus récemment des auteurs ont essayé, en se basant sur l'approche initiale de la projection des segments de la sphère de Gauss, d'injecter des informations statistiques à priori. Parmi ces auteurs il y en a [Collins et Weiss, 1990] qui partent du fait que les segments sont déjà classés préalablement, et essaient ensuite de trouver la localisation du point de fuite en fonction de modèles statistiques.

D'autres considérations peuvent être le comptage préalable des points de fuite, comme par exemple chez [Antone et Teller, 2000], ou bien de réduire l'espace d'accumulation sur la sphère de Gauss en donnant préalablement l'orientation approchée des images, ce qui a été traité par Shuffelt [Shufelt, 1999] dans le but d'une application sur des images aériennes obliques. Il est évident que dans un cas général il est difficile de faire des hypothèses sur

le nombre de points de fuite d'une scène. De plus, dans notre cas, on cherche justement à définir l'orientation des images avec l'aide des points de fuite, donc une méthode où l'on doit connaître préalablement l'orientation approchée des images ne peut être retenue.

[Boulanger *et al.*, 2006] a proposé une variante de la sphère de Gauss mais avec une amélioration importante du temps de calcul. Son algorithme travaille aussi sur une sphère discrétisée, mais dans sa démarche il propose de travailler sur un seul hémisphère au lieu de toute la sphère, ainsi que différentes façons de discrétiser de manière optimum les cellules d'accumulation. Dans [Zhang et Kosecka, 2002] est présentée une méthode de détection des points de fuite et de classification des différentes familles de segments, sans la nécessité d'avoir les paramètres d'étalonnage de la caméra. Cette méthode utilise l'algorithme espérance-maximisation (EM), et fait l'hypothèse que dans les objets fabriqués par l'homme la majorité des points de fuite sont dans trois directions orthogonales deux par deux.

[Van den Heuvel, 1998] utilise les normales aux plans d'interprétations. Le premier point de fuite peut être extrait de manière manuelle. En supposant qu'il n'y a que deux autres points de fuite perpendiculaires à la direction du premier détecté, des hypothèses sur l'orthogonalité des points de fuite sont faites. Toutefois, notre expérience des scènes urbaines montre que l'on peut très bien avoir plus de trois points de fuite. Par exemple dans la Figure 4.4, la scène comporte 5 points de fuite.

Au total : les approches sur la sphère de Gauss entraînent des changements de paramètres et des stratégies d'accumulation souvent coûteuses (discrétisation de la sphère de Gauss, etc...) ainsi que la nécessité de connaître les paramètres intrinsèques de la camera. Par contre elles sont généralement très précises. Aujourd'hui, elles utilisent essentiellement des méthodes probabilistes et nécessitent des connaissances a priori sur la scène.

*Travail dans l'espace de la transformée de Hough*

Issue d'un brevet déposé par Hough en 1962 [Hough, 1962], pour reconnaître des trajectoires de particules dans des images de chambres à bulles, cette méthode a un champ d'applications très étendu. Elle consiste, schématiquement, à se placer dans un espace de paramètres qui décrivent la forme à reconnaître, et à rechercher dans cet espace des points d'accumulation. Le principe général de la transformée de Hough est d'établir une projection entre l'espace de l'image et un espace de paramètres representatif de la forme recherchée. En d'autres termes, on transforme l'image dans l'espace des paramètres et on identifie la courbe dans cet espace. L'espace de projection des paramètres est borné, ce qui permet de construire un accumulateur discret, constitué de "cases" dont le poids augmente en fonction du nombre de courbes qui y passent. Les points d'accumulation sont alors des maxima locaux sur cet accumulateur. Ces accumulateurs sont parfois de taille très grande, lourds à manipuler, ce qui a justifié des méthodes de divisions hiérarchiques pour discrétiser ces accumulateurs. [Lutton, 1990]. La transformée de Hough est un cas particulier de la transformée de Radon datant de 1917 [Radon, 1917] .

Par exemple dans la transformée de Hough, chaque segment de l'espace image est converti en un point de coordonnées $\rho$ et $\theta$ (coordonnées polaires définies Figure **??**). L'ensemble des

segments passant par un point donné forme donc une sinusoïde d'axe parallèle à l'axe des $\theta$, et dans cet espace, une droite est représentée par un point.

Citons ici aussi [Kender, 1979], qui a proposé une amélioration conséquente de la méthode de Hough, d'abord à partir des cercles de Thalès que nous détaillerons plus loin 5, puis en appliquant une inversion transformant ces cercles en droites, ce qui avec les moyens de calcul cette époque était bien plus opérationnel. Il cherchait les points de fuite pour décrire la géométrie de surfaces à partir de leur texture et se basait sur des méthodes d'accumulation.

En 1989, [Quan et Mohr, 1989] ont proposé une méthode basée sur la méthode de [Barnard, 1983]. Cette méthode utilise la transformée de Hough comme accumulateur sur la sphère de Gauss. En 1994, Lutton et al. [Lutton *et al.*, 1994] proposent une méthode fondée sur la transformé de Hough [Hough, 1962] et sur la sphère de Gauss. La différence des deux méthodes est l'échantillonnage différent de l'espace des paramètres. La méthode de [Lutton *et al.*, 1994] est plus résistante aux bruits sur les segments, plus rapide que [Quan et Mohr, 1989] car elle détecte les points de fuite en une seule itération, tandis que l'algorithme de [Quan et Mohr, 1989] nécessite plusieurs itérations. L'inconvénient majeur est que ces méthodes citées cherchent seulement les trois points de fuite qui correspondent à trois directions orthogonales. Comme il sera dit plus tard, c'est un inconvénient sérieux, car le nombre de points de fuite dans une image ne peut jamais préalablement être défini (voir Figure 4.4). Tuytelaars [Tuytelaars *et al.*, 1998] introduit en 1998 une méthode interactive basée également sur la transformée de Hough, sous le nom de "Cascade Hough Transform". On peut encore rajouter ici [Quan et Mohr, 1989].

*Travail dans l'espace image*

Dernière catégorie, certains algorithmes n'utilisent pas d'espace de projection ou d'espace fini d'accumulation, et travaillent directement dans l'espace 2D de l'image. La majorité de ces algorithmes sont basés sur l'intersection des segments de l'image. Par exemple, il y a les algorithmes [Brillault-O'Mahoney, 1991], [Rother, 2002]qui testent l'intersection de tous les segments de l'image deux par deux (Figure 4.5).

Bien que simples à mettre en oeuvre, ces algorithmes ont une complexité de calcul très élevée qui augmente de manière combinatoire avec le nombre de segments. [Rother, 2002] propose une méthode très simple pour la détection des points de fuite. Sa méthode consiste à effectuer l'intersection de tous les segments de l'image deux par deux. Cette étape est suivie d'une étape de vote majoritaire. Ces approches basées sur l'intersection des segments de l'image ont pour elles l'avantage d'être simples, car elles travaillent dans un espace à 2 dimensions et ne nécessitent pas de modèles mathématiques complexes. Leurs défauts principaux sont de ne pas pouvoir traiter de manière efficace les points de fuite qui partent vers l'infini pour lesquels, clairement, cette intersection des segments est mal définie. En général on parle d'intersection en sifflet. De même pour la méthode de [Cantoni *et al.*, 2001], qui a comme plus gros défaut l'impossibilité de détecter les points de fuite en dehors de l'image, et ainsi donc les points de fuite à l'infini (cas très courant). D'autre articles comme [Almansa *et al.*, 2003] utilisent des principes mathématiques poussés de regroupement, comme celui de Helmoltz. Cette méthode ne nécessite aucune information sur les paramètres intrinsèques de l'image, mais utilise des modèles probabilistes complexes (Figure 4.6).

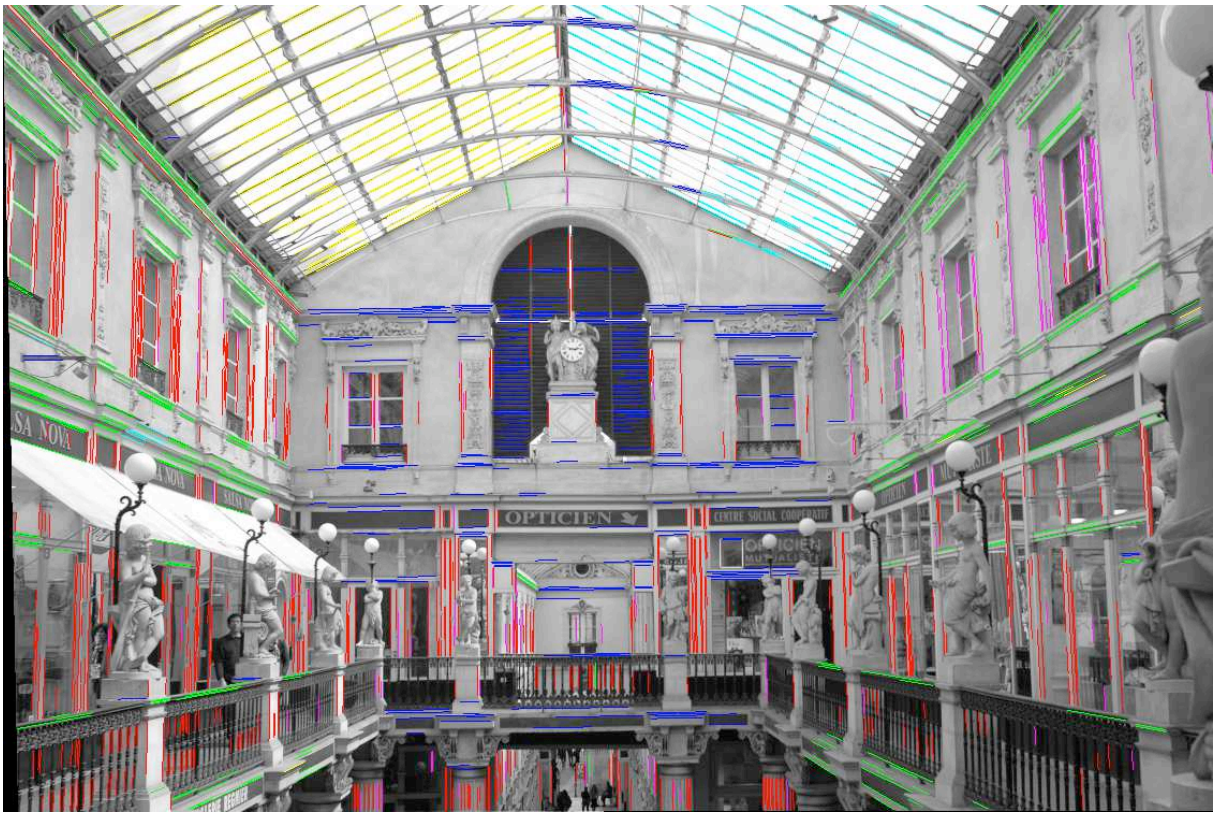



**Fig. 4.4.** Image du célèbre "Passage Pommeraye" à Nantes immortalisé par Jacques Denis dans un film. Cette scène comporte cinq points de fuite. Résultats de classification des différentes familles de segments correspondant aux familles de points de fuite. Deux familles de segments parallèles sont situées sur le toit en verre (en jaune et cyan).

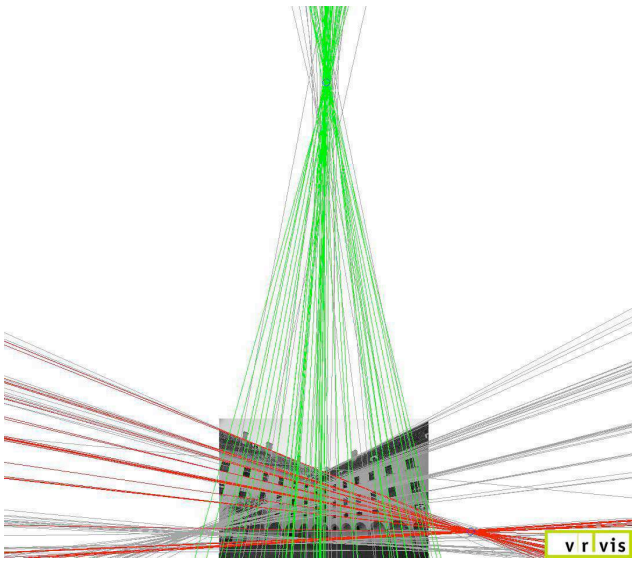



**Fig. 4.5.** Illustration de la méthode de [Rother, 2002] utilisée dans le projet MetropoGIS [Bauer *et al.*, 2002]

*Une méthode atypique de détection des points de fuite*

Une approche qui mérite d'être citée est celle de [Stentiford, 2006], car contrairement aux méthodologies classiques de détection de points de fuite celle-ci n'utilise pas comme point

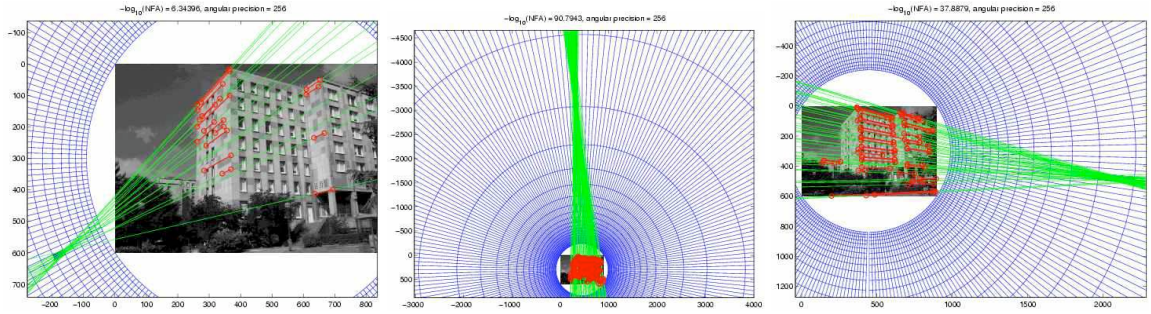


**Fig. 4.6.** Résultats de la détection des points de fuite avec la méthode de Almansa, figure tirée de [Almansa *et al.*, 2003]

de départ les segments de l'image. Cette approche originale est basée sur l'attention visuelle humaine. En effet il est supposé que dans une image contenant une perspective marquée, l'attention humaine se focalise sur le point de fuite, quelques résultats de sa détection sont illustrés dans la Figure 4.7.

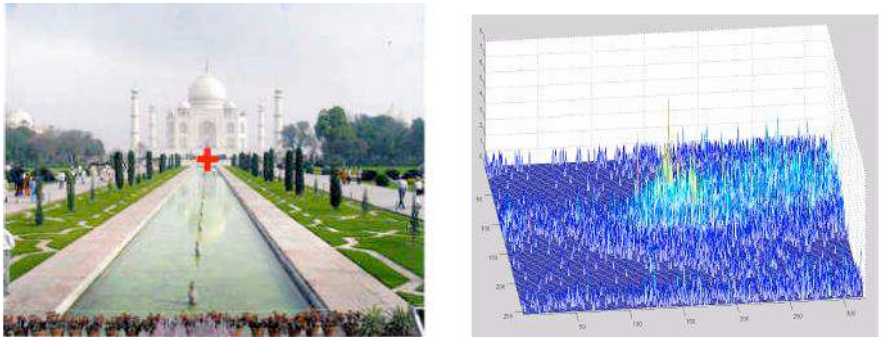


**Fig. 4.7.** Résultat de la détection du point de fuite avec l'algorithme de [Stentiford, 2006]. L'image de droite résulte du suivi géométrique de l'œil de l'observateur.

Cette méthodologie est basée sur un modèle d'attention qui consiste à détécter les éléments qui accrochent l'attention dans un premier regard sur l'image. Un des éléments accrocheurs est la présence d'une symétrie due à la perspective. A partir de son modèle d'attention, et grâce à un suivi des mouvements de l'œil, il crée un histogramme d'attention en fonction des coordonnées des pixels. Le pic de cet histogramme correspond à l'emplacement du point de fuite (voir Figure 4.7). Le défaut majeur de sa méthode, outre le fait qu'elle exige un observateur et qu'elle n'est donc pas automatique, est qu'elle ne peut traiter les scènes contenant qu'un seul point de fuite.

En fonction de la méthodologie employée pour la détection des points de fuite et de l'espace employé, nous pouvons présenter une classification générale de ces algorithmes. Deux grandes familles se distinguent : les méthodologies utilisant des algorithmes d'accumulation et de vote majoritaire, et les algorithmes employant des techniques robustes de classification et d'estimation, telles [Fischler et Bolles, 1981]. Dans le tableau 4.3 est présenté un résumé

des méthodes citées en fonction de leur espace de travail et du type d'algorithme employé.

| Espace de travail | Sphère de Gauss | Transformée de Hough | Image |
|---|---|---|---|
| Méthode d'accumulation | [Barnard, 1983]<br>[Magee et Aggarwal, 1984]<br>[Shufelt, 1999] | [Kender, 1979]<br>[Lutton *et al.*, 1994]<br>[Quan et Mohr, 1989]<br>[Tuytelaars *et al.*, 1998]<br>[Boulanger *et al.*, 2006]<br>[Cantoni *et al.*, 2001] | [Rother, 2002]<br>[Almansa *et al.*, 2003]<br>[Brillault-O'Mahoney, 1991] |
| Estimateur par tirage (type RANSAC) | [Antone et Teller, 2000]<br>[Zhang et Kosecka, 2002] | ..<br>..<br>.. | [Aguilera et Gomez Lahoz, 2005]<br>[Almansa *et al.*, 2003] |

**Tableau 4.1.** Classification générale des algorithmes de détection des points de fuite existants.

## 4.4 Applications des points de fuite en imagerie

Les applications de la détection des points de fuite sont très nombreuses. Nous essayerons de lister les principales applications citées actuellement dans la littérature.

### 4.4.1 Calcul des paramètres intrinsèques de la caméra

Une des applications les plus répandues des points de fuite est le calcul des paramètres intrinsèques d'une caméra inconnue à partir d'une image, c'est-à-dire sa calibration. Dans l'histoire de la géométrie projective, ce problème est connu sous le nom de perspective inverse. Avant le XX$^{\text{ème}}$ siècle, deux personnes au moins se sont intéressées à ce problème et ont essayé d'y apporter une solution. Brook Taylor, plus connu pour ses développements limités, a publié dans son Traité de "New principles of linear perspective" de 1715, et Jean-Henri Lambert qui donna, dans "die Free Perspective" de 1759 et "de Note et Additions" de 1769 une série de problèmes résolus pour la restitution des paramètres internes d'une caméra. [Benallal, 2002] dans sa thèse explique avec beaucoup de précision ces méthodes de calibrations basées sur les travaux de ces deux illustres géomètres du $XVIII^{\text{ème}}$ siècle. Plus récemment d'autres auteurs ont proposé des méthodes de calibration en se basant sur les points de fuite, on peut citer [Caprile et Torre, 1990], [Kanatani et Onodera, 1990], [Beardsley et Murray, 1992], [Cipolla *et al.*, 1999], [Xie *et al.*, 2004], [Wilczkowiak *et al.*, 2005], [Grammatikopoulos *et al.*, 2006]. Il est important de noter qu'afin de pouvoir procéder à la calibration avec l'aide des points de fuite, il faut avoir trois points de fuite orthogonaux deux à deux (Figure 4.6). Depuis lors, le nombre d'applications basées sur cette approche n'a cessé d'augmenter. Par exemple, cela nous permet d'obtenir des informations sur une image ancienne, et de pouvoir calculer sa focale, son PPA [1], sa distorsion, etc... Ainsi, se focalisant sur tel ou tel de ces paramètres, de nombreux auteurs se sont penchés sur la question, notamment dans la sphère de préservation du patrimoine, [Karras et Petsa, 2001], [Fangi *et al.*, 2001], [Brauer-Burchardt, 2001].

1. PPA : point principal d'autocollimation

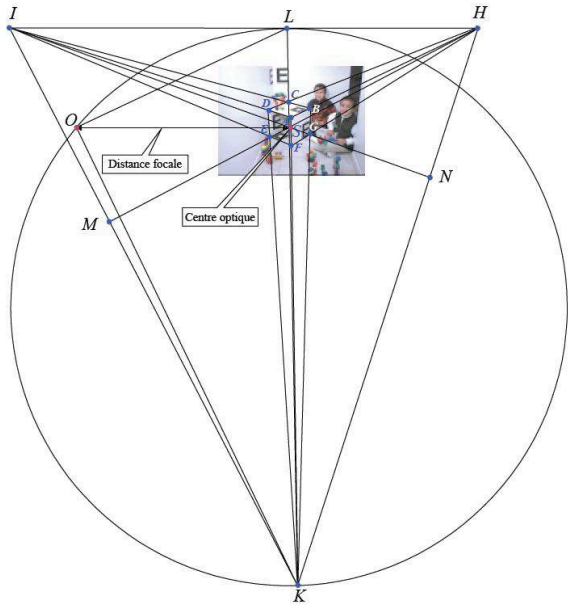



**Fig. 4.8.** Illustration de la méthode géométrique de Brook Taylor, image tirée de la thèse de [Benallal, 2002].

### 4.4.2 Estimation de l'orientation de l'image

Une des applications les plus importantes des points de fuite est l'estimation de l'orientation de l'image avec l'aide d'une seule prise de vue. En effet avec la localisation des points de fuite il est possible de calculer l'orientation de l'image par rapport à l'objet. Quand on dit objet, on pense surtout aux objets fabriqués par la main de l'homme, comme les bâtiments, fenêtres, etc. C'est cette application qui nous intéresse. Le modèle mathématique employé est très simple. Dans cette modélisation nous partons du fait que la calibration de la caméra est à priori connue. Le système de la camera est défini par les trois axes $(X_{cam},\ Y_{cam},\ Z_{cam})$, le plan focal est à $Z_{cam} = F$, $F$ étant la distance focale. Le système d'axes terrain est défini par $(X_T,\ Y_T,\ Z_T)$. Dans ce système l'axe $Y_w$ est parallèle à l'axe vertical physique. Les deux systèmes ont la même origine $C$ qui est le centre optique de la caméra. $v$ est l'endroit où se localise le point de fuite vertical sur l'image. De même pour $h$ qui correspond au point de fuite horizontal sur l'image. Depuis le centre de la caméra $C$ les vecteurs $\overrightarrow{V_{ver}}$ et $\overrightarrow{V_{hor}}$ sont tracés. Ces deux vecteurs sont définis dans le système d'axes de la caméra. Par définition $\overrightarrow{V_{ver}}$ est perpendiculaire à $\overrightarrow{V_{hor}}$. On peut aisément déduire le troisième axe par produit vectoriel. Nous nommerons ce vecteur $\overrightarrow{V_{hv}}$. On note que cette définition du troisième axe est utilisée quand on n'a que deux points de fuite (par exemple la photo d'une façade prise de face, Figure 4.10.a). Si un bâtiment est photographié d'une manière telle que les trois points de fuite orthogonaux sont identifiés, alors nous avons directement le vecteur $\overrightarrow{V_{hv}}$ (Figure 4.10.b). Maintenant il est très simple de calculer la matrice rotation de passage du système d'axes caméra au système d'axes terrain. La rotation de passage du système caméra au système terrain $R_C^T$ sera alors :

$$R_C^T = \left[\left|\overrightarrow{V_{ver}}\right| \left|\overrightarrow{V_{hor}}\right| \left|\overrightarrow{V_{hv}}\right|\right] \tag{4.1}$$

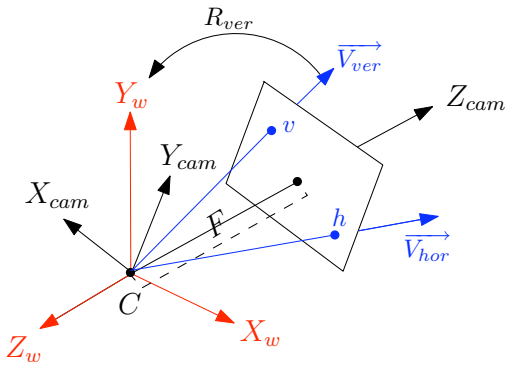



**Fig. 4.9.** Définition des différents système d'axes.

(a) (b)

**Fig. 4.10.** a) Exemple d'un bâtiment avec deux points de fuite. b) Exemple d'un bâtiment avec trois points de fuite.

Si seul le point de fuite vertical est connu c'est-à-dire $\overrightarrow{V_{ver}}$ on peut seulement définir la matrice de rotation qu fait le passage entre les vecteurs $\overrightarrow{V_{ver}}$ et $\overrightarrow{Y_T}$ (rappel : $Y_T = [0\ ,1\ ,0]^T$). Pour cela il faut d'abord définir l'axe de rotation $\overrightarrow{\omega}$ ainsi que l'angle de rotation $\theta$ : $\overrightarrow{\omega} = \overrightarrow{V_{ver}} \otimes \overrightarrow{Y_T}$, après simplification et normalisation $\overrightarrow{\omega} = [\frac{V_z}{d}\ ,0\ ,\frac{-V_x}{d}]$, où $d = \sqrt{V_z^2 + V_x^2}$ , $\theta = \arccos(\overrightarrow{V_{ver}} \cdot \overrightarrow{Y_T})$, après simplification, $\theta = \arccos(V_y)$. En utilisant la formule d'Olinde-Rodrigues nous obtenons la matrice de rotation $R_{ver}$ :

$$R_{ver} = I\cos\theta + \sin\theta\,[\omega]_\times + (1-\cos\theta)\omega\ {}^t\omega. \tag{4.2}$$

L'estimation de la rotation à l'aide des points de fuite est très utilisée en milieu urbain. La littérature en ce domaine est très riche, ce qui montre le potentiel de cet emploi dans de tels environnements. Citons : [Lee *et al.*, 2002], [Antone et Teller, 2000], [Teller *et al.*, 2001], [Bauer *et al.*, 2002], [Georgiadis *et al.*, 2005], [Schindler et Joachim, 2003], [Kosecka et Zhang, 2002].
D'autres auteurs comme [Lobo et Dias, 2003] ont couplé les données issues d'un système inertiel miniaturisé (MEMS-IMU) avec les informations extraites de l'images en utilisant des points de fuite.

Une application, qui dérive directement de l'estimation de l'orientation avec l'aide des points de fuite, est la rectification de l'image selon les directions des points de fuite détectés. Ainsi, pour un faible coût on peut obtenir des "pseudo-orthoimages". La littérature est très riche dans ce domaine, que ce soit en photogrammétrie ou en vision par ordinateur. L'intérêt

dans le domaine de la photographie est majeur : par exemple chez Kodak [Gallagher, 2005], des recherches ont été faites pour remettre à l'horizontale des images penchées, en utilisant les points de fuite (Figure 4.11).

(a) (b)

**Fig. 4.11.** Alignement de l'image sur la verticale : ([Gallagher, 2005] a) Image de départ b) Image corrigée.

[Criminisi *et al.*, 1999] utilise les points de fuite, afin de reconstruire une scène juste à partir d'une seule image. Ses travaux ont eu une grande importance dans le domaine de l'art (voir Figure 4.12).

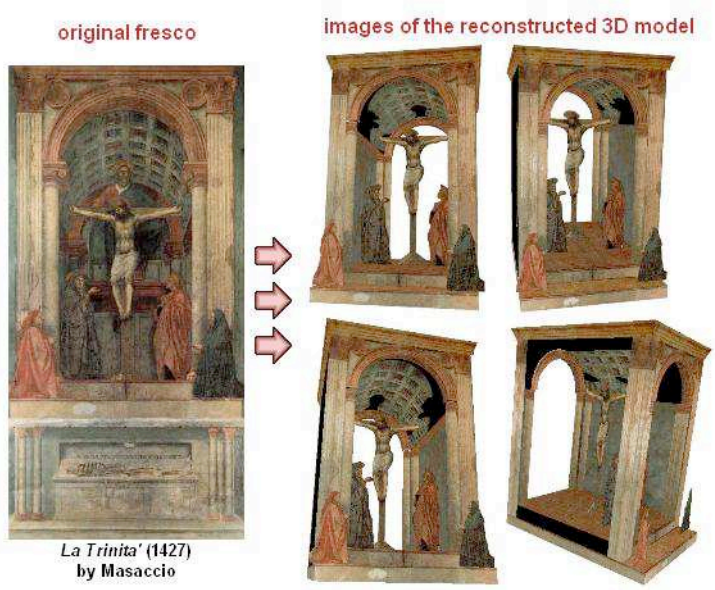



**Fig. 4.12.** Illustration d'une reconstruction 3D de l'image à partir d'une seule image, selon les travaux de [Criminisi *et al.*, 1999].

[Boulanger *et al.*, 2006] se sert des points de fuite afin de créer de manière automatique différents point de vue à partir d'une seule image. Dans sa démarche, les points de fuite sont utilisés pour étalonner la caméra.

Enfin nous terminons ce paragraphe avec quelques illustrations (Figure 4.13) d'images rectifiées avec l'aide des points de fuite détectés par les algorithmes présentés dans les chapitres 5 et 6.

**Fig. 4.13.** Rectification d'image à l'aide des points de fuite détectés à l'aide des algorithmes présentés dans cette thèse (chapitres 5 et 6)

.

### 4.4.3 Autres applications

*Détection de rectangles*

Enfin la détection des points de fuite peut aussi permettre de classifier les différentes familles de segments en fonction des directions. Cette classification permet par la suite la détection d'autres éléments structurés comme par exemple les rectangles. Ainsi [Shaw et Barnes, 2006], [Kosecka et Zhang, 2005] utilisent les points de fuite pour la détection d'éléments rectangulaires. Dans [Kosecka et Zhang, 2005], les rectangles sont ensuite appariés et mis en correspondance, ce qui permet de s'affranchir de points de liaison pour l'estimation de l'orientation, voir Figure 4.14. [Cole et Barnes, 2008] emploient les points de fuite pour

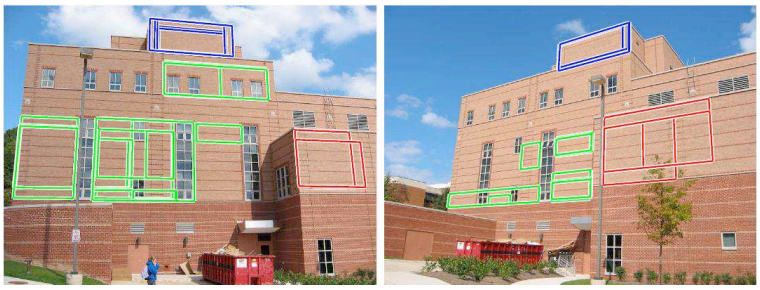


**Fig. 4.14.** Exemple de détection et d'appariement des rectangles. Chaque famille de couleurs correspond aux rectangles appariés.

la vision d'un robot qui se déplace dans des environnements structurés.

*Détection des éléments routiers, aide à la navigation*

Les points de fuite ont une application directe dans la segmentation du plan de la route. Ceci permet d'assister des véhicules en vue de les rendre autonomes, et de fournir une aide à la

navigation. Les travaux publiés traitent des images de routes avec des marquages bien lisibles ce qui permet une détection optimale des segments qui vont servir ensuite à la détection des points de fuite, on peut citer notamment [Simond et Rives, 2004], [Nieto *et al.*, 2008], [Macek *et al.*, 2004]. Dans un contexte plus général et plus difficile, des algorithmes [Alon *et al.*, 2006], [Kong *et al.*, 2009] détectent les points de fuite sur des routes sans aucun marquage (par ex : les routes en zone désertique ou en campagne (voir Figure 4.15)).

(a) (b)

**Fig. 4.15.** Exemple de détection des points de fuite sur : a) des routes de ville avec marquage [Macek *et al.*, 2004] b) sur des routes de campagne [Kong *et al.*, 2009]

L'utilisation des points de fuite est aujourd'hui de plus en plus fréquente dans les logiciels d'édition de photos comme Photoshop CS2 (voir Figure 4.16). Avec l'aide de ce nouvel outil, il est possible d'insérer des motifs, tout en respectant la perspective. Actuellement dans Photoshop la détection des lignes et du point de fuite se fait de manière manuelle, mais il est clair qu'elle sera amenée à devenir automatique dans les années à venir.

**Fig. 4.16.** Résultats au moyen de l'outil "Point de fuite" de Photoshop CS2, qui permet de dupliquer, de peindre et de coller des éléments respectant automatiquement la perspective de l'image.

## 4.5 Conclusion

Dans ce chapitre nous avons passé en revue les différentes méthodes de détection des points de fuite. Nous avons vu que la littérature est très abondante sur le sujet, ce qui en caractérise l'intérêt pour beaucoup d'applications.

Nous avons vu que la majorité d'entre elles, indépendamment de l'espace de travail et de projection, se terminent par une fastidieuse étape d'accumulation. Cette évidence provient de la non exactitude des points ou des repérages de points dans l'image, qui ne peut être completement corrigée par le modèle géométrique employé. Toutefois, il est désormais possible de s'en affranchir.

Dans l'étude présente, nous avons mis en place un algorithme de classification et de détection, inspirée du Ransac classique. L'avantage de celui-ci est qu'il peut très simplement s'adapter à plusieurs espaces de travail. Ici nous l'avons utilisé successivement dans l'espace image et l'espace de la sphère de Gauss. Dans les deux cas, nous verrons que le problème de classification des différentes gerbes perspectives revient tout simplement à classer des primitives géométriques, comme des cercles chapitre 5 et des plans chapitre 6. Ce qui permettra de trouver les points de fuite de manière très rapide.

L'autre objectif que nous avons recherché est de pouvoir maitriser, de manière simple, l'incertitude sur le point de fuite en fonction de la qualité de détection des segments. Car pour un ré-emploi des points de fuite dans d'autres processus de calcul, par exemple le calcul d'une matrice de rotation, il est important de savoir avec quelle précision celle-ci est obtenue.

Nous allons donc désormais examiner nos deux nouvelles approches de ce problème très ancien, revisité avec les outils actuels d'estimation robuste.

# 5

# Détection des points de fuite dans l'espace image : méthode des cercles

## 5.1 Introduction

Un algorithme entièrement nouveau et automatique de détection de points de fuite est présenté dans ce chapitre. Cette approche travaille dans l'espace à 2 dimensions de l'image, dont la géométrie est décrite dans le paragraphe 5.2.

Afin de pouvoir extraire les nuages de points entremêlés, un algorithme de tri basé sur la méthode d'estimation robuste de type RanSac [Fischler et Bolles, 1981] est présenté dans la section 5.3. Cette estimation robuste est ensuite raffinée par une propagation d'incertitude exploitant les variances individuelles de chaque segment qui fait l'objet de la section 5.4.

Un avantage important vis à vis des autres méthodes est, d'une part, la non nécessité de prise en compte de certains paramètres intrinsèques de la caméra, en particulier de l'orientation interne, pour la seule extraction des points de fuite, et d'autre part le mécanisme de propagation d'incertitude permettant de qualifier le point de fuite. L'autre avantage majeur de cet algorithme est que l'étape de classification des différentes directions des segments se fait en même temps que la détermination du point de fuite.

Cet algorithme, bien que travaillant dans l'espace image, permet sans aucune difficulté de détecter même les points de fuite rejetés à l'infini. Ce qui lui permet de rivaliser avec les méthodes utilisant un espace d'accumulation intermédiaire, du type de la sphère de Gauss. Les performances de l'algorithme sont évaluées dans la partie 5.5.

Sa comparaison avec d'autres méthodes de détection des points de fuite fera l'objet du paragraphe 6.8.

## 5.2 Géométrie projective, théorème de Thalès et points de fuite

Un théorème de géométrie projective (Chasles-Steiner) permet de montrer que la recherche de points de fuites revient à détecter des cercles passant par un point fixe dans une image. Le

théorème de Chasles-Steiner est une généralisation du théorème de Thalès pour l'ensemble des coniques. Une transformation permet d'associer à chaque segment de l'image un point qui devra contribuer à la formation de cercles si ce segment contribue à un même point de fuite. Dans cette partie, ce théorème de la géométrie projective est rappelé. Par la suite son emploi dans la recherche des points de fuite est analysé en détail. L'emploi de cette géométrie simple et élégante a déjà été partiellement utilisé dans [Kender, 1979] et [Brauer Burchardt et Voss, 2000].

### 5.2.1 Description géométrique d'une conique

La description d'une conique en géométrie projective repose entièrement sur le théorème fondamental de Chasles-Steiner qui peut être énoncé succinctement : "Une homographie entre deux faisceaux de droites définit une conique, et réciproquement" [Sidler, 2000], [Semple et Kneebone, 1952].

**Définition 1** *Dans un plan projectif, l'ensemble des droites qui passent par un point donné $A$ s'appelle un faisceau de droites, qu'on désigne par $A^*$.*

**Théorème de Chasles-Steiner 1** *Soient $A$ et $B$ deux points distincts et $\alpha : A^* \rightarrow B^*$ une homographie entre les faisceaux de droites $A^*$ et $B^*$. On suppose que $\alpha$ n'est pas une projection et on désigne par $O$ son centre d'homographie. Dans ces conditions, pour toute droite $d$ de $A^*$, le lieu du point $d \bigcap \alpha(d)$ est une conique, tangente en $A$ à la droite $AO$ et en $B$ à la droite $BO$.*

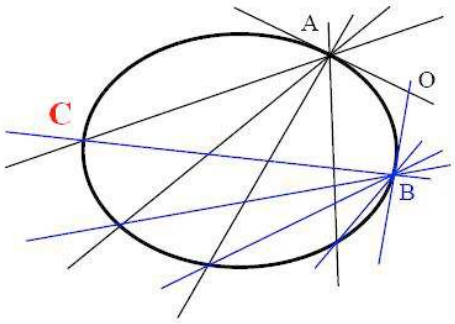



**Fig. 5.1.** Illustration du théorème Chasles-Steiner. Si $A$ coupe $B$ avec un angle constant, leur intersection définit le point $C$ de la conique.

Ce beau théorème permet la construction des coniques dans l'espace projectif. Si une notion d'angle est rajoutée dans cette définition, la conique sera ramenée dans un espace euclidien. Et si les faisceaux en correspondance homographique se coupent avec un angle droit, cette conique sera un cercle : on retrouve ici le théorème de Thalès bien connu.

### 5.2.2 Géométrie de l'image et points de fuite

Si $S_1$, $S_2$, $S_3$ sont trois droites parallèles dans l'espace réel, leurs images après une projection conique sont 3 droites concourantes qui se coupent sur le point $P$. Ce point est

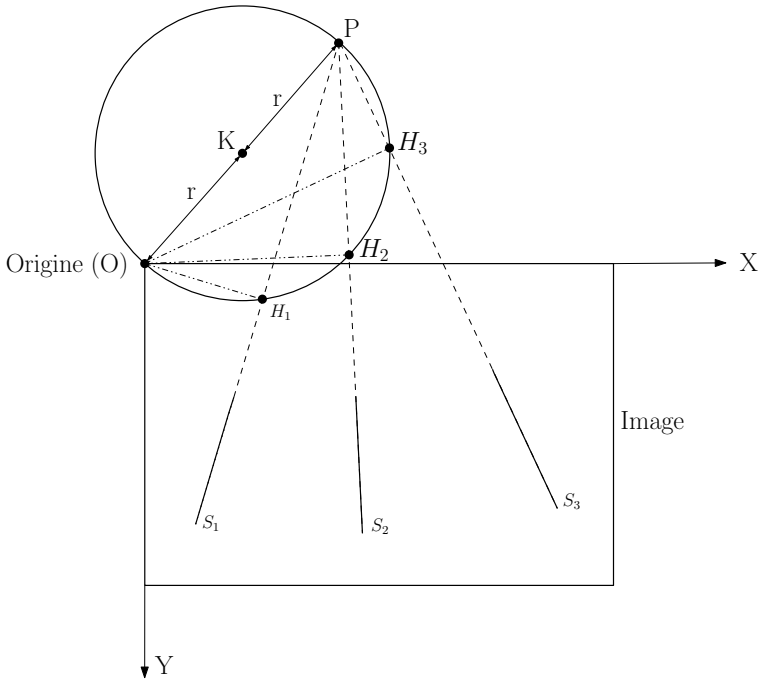



**Fig. 5.2.** Configuration géométrique du théorème Chasles-Steiner et du point de fuite.

conventionnellement connu sous le nom de point de fuite. Maintenant un nouveau faisceau de droites avec pour centre $O$ (l'origine de l'image) est construit de façon que, une par une, les droites de ce faisceau se coupent avec un angle droit les droites $S_1$, $S_2$, $S_3$ selon les points $H_1$, $H_2$, $H_3$. Suivant le théorème de Chasles-Steiner, $H_1$, $H_2$, $H_3$ et l'origine $O$ définissent un cercle (Figure 5.2 et Figure 5.3).

Afin de pouvoir déterminer le point P (le point de fuite) il suffira juste de déterminer les paramètres du cercle. Chercher les points de fuite qui peuvent correspondre à l'intersection de segments dans les images revient à présent à détecter des objets très simples : des cercles (Figure 5.4).

L'exploitation du théorème de Thalès pour la recherche de points de fuite a déjà été proposée par Brauer and Voss [Brauer Burchardt et Voss, 2000], mais sans traiter le problème de l'extraction des différents cercles entremêlés, et sans analyse de variance. La présente étude a été complétée dans ce sens. Par ailleurs cette même géométrie a été proposée par [Kender, 1979], à laquelle il rajoutait une inversion qui, à cette époque, en transformant les cercles en segments de droites, facilitait beaucoup le calcul. Nous n'avons pas retenu cette partie de son approche, qui aujourd'hui n'est pas déterminante.

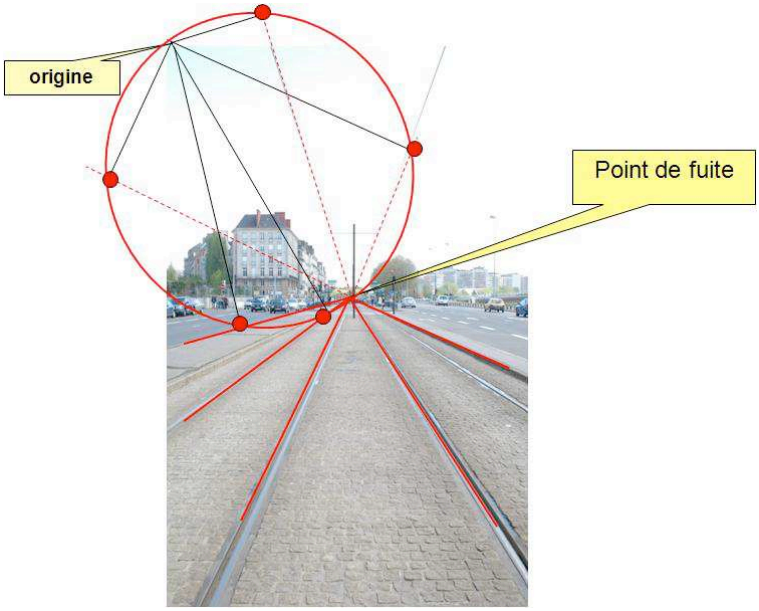
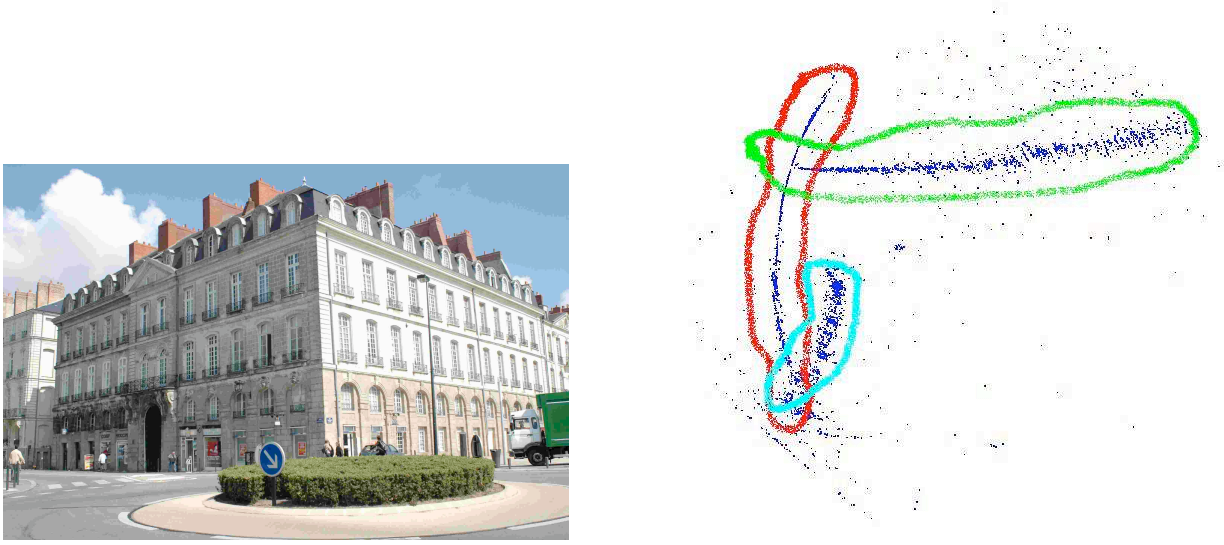



**Fig. 5.3.** Illustration du théorème Chasles-Steiner et de Thalès sur une image réelle.



(a) (b)

**Fig. 5.4.** a) Image d'un bâtiment qui contient trois points de fuite, un point de fuite pour les verticales et deux points de fuite pour des horizontales. b) les cercles correspondants (dont les points associés ont été entourés manuellement) sont bien au nombre de trois.

## 5.3 Algorithme et mise en œuvre

Dans cette partie, toutes les étapes de la détection des points de fuite sont détaillées. Ainsi les concepts vus précédemment trouvent leur application. Le schéma global de l'algorithme est présenté dans la Figure 5.5.

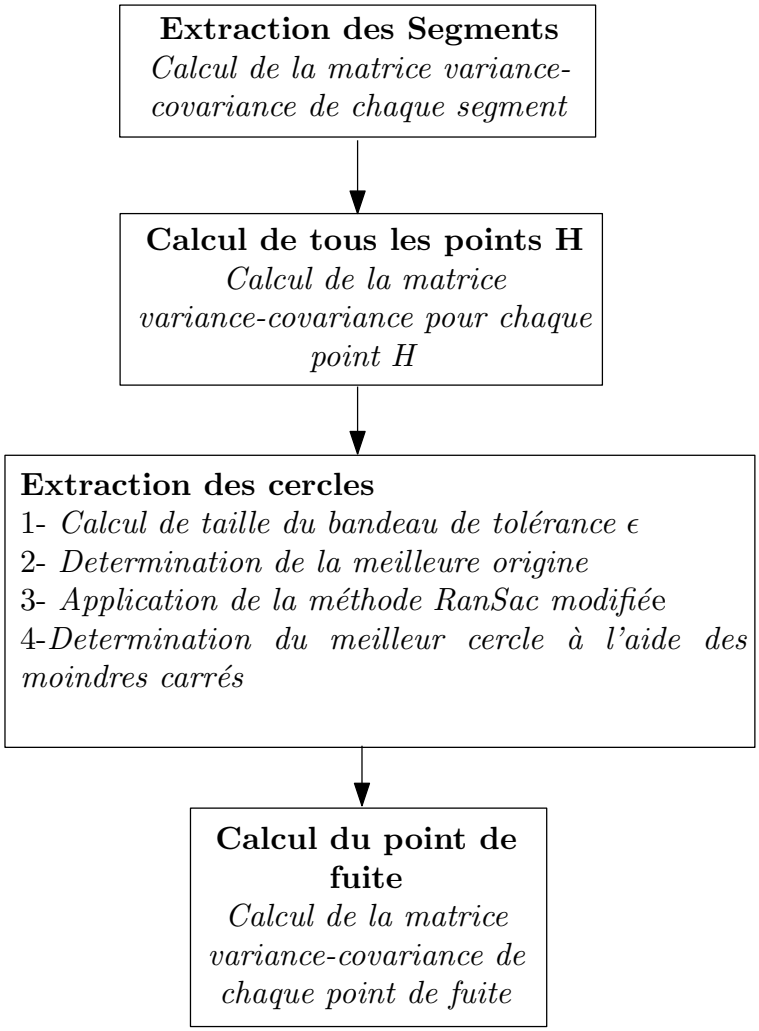



**Fig. 5.5.** Les différentes étapes de la mise en œuvre. En italique, sont mentionnées les différentes sous-étapes nécessitées par le besoin de connaître les erreurs et leur propagation dans les calculs.

.

### 5.3.1 Extraction automatique des segments de l'image

La première étape des algorithmes de détection de point de fuite est l'extraction des segments de l'image. Celle-ci est décrite dans la section 2.3.2. Un des principaux résultats de cette étape est :
1. l'extraction de tous les segments de l'image,
2. la matrice variance-covariance de chaque segment.

Tous les segments sont détectés dans le système d'axes présenté dans la Figure 5.6.

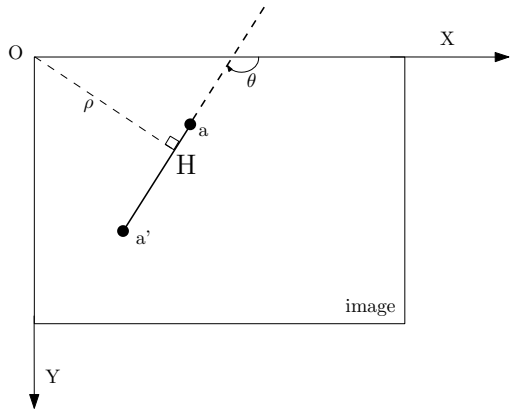



**Fig. 5.6.** Définition du système image et du segment $a - a'$

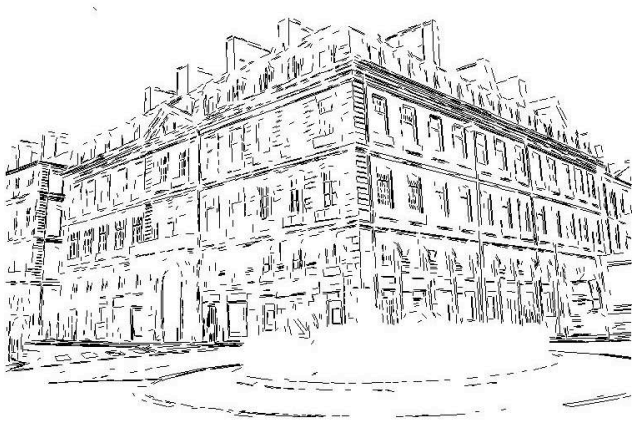


**Fig. 5.7.** Résultats de détection de segments un bâtiment à Nantes. 5800 segments ont été extraits.

### 5.3.2 Calcul des points $H_i$

L'équation polaire (encore appelée équation normale) de la droite support de chaque segment détecté est la suivante :

$$\rho = -x \sin\theta + y \cos\theta \tag{5.1}$$

Le point $H_i$ est le pied de la droite perpendiculaire au segment considéré issue de $O$ qui est défini par :

$$H_i = \begin{bmatrix} -\rho_i \sin\theta_i \\ \rho_i \cos\theta_i \end{bmatrix} \tag{5.2}$$

Il est important de pouvoir propager la matrice variance-covariance de chaque segment sur son point $H_i$ correspondant. En utilisant la loi générale de la propagation d'erreurs pour les fonctions non linéaires, et en se basant sur le théorème de Taylor :

$$\Sigma_H = J.\Sigma_{\theta,\rho}.J^T \tag{5.3}$$

Où $J$ est la matrice Jacobienne qui contient les dérivées de $H_i$ selon $\theta$ et $\rho$. $\Sigma_{(\theta,\rho)}$ est la matrice d'incertitude du segment en fonction de $\rho$ et $\theta$. La matrice variance-covariance sur chaque point $H_i$ est obtenue ainsi :

$$\Sigma_{H_i} = \begin{bmatrix} -\rho_i \cos\theta_i & -\sin\theta_i \\ -\rho_i \sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} \sigma^2_{i(\theta)} & \sigma_{i(\theta,\rho)} \\ \sigma_{i(\theta,\rho)} & \sigma^2_{i(\rho)} \end{bmatrix} \begin{bmatrix} -\rho_i \cos\theta_i & -\sin\theta_i \\ -\rho_i \sin\theta_i & \cos\theta_i \end{bmatrix}^T \tag{5.4}$$

On peut illustrer de manière classique les matrices variance covariance des points $H_i$ par leurs ellipses d'erreurs. La Figure 5.8 représente les ellipses d'erreur obtenues à partir des segments trouvés sur la Figure 5.4.a et des points $H_i$ correspondants.

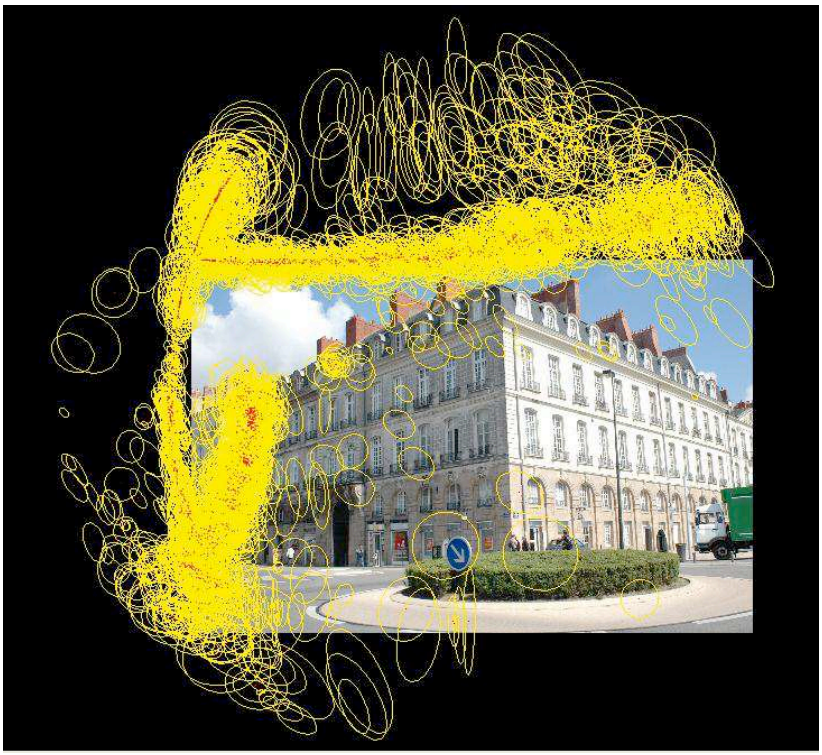


**Fig. 5.8.** Ellipses d'erreur à 95% de confiance montrant l'incertitude de détermination des points $H_i$, fonction de l'efficacité de l'outil d'extraction automatique des segments, à partir de l'image présentée en Figure 5.4.a

### 5.3.3 Extraction des cercles

Il y a ici deux problèmes distincts à traiter :

1. Quelle modélisation choisir pour le meilleur cercle passant par un nuage de points ?
2. Comment extraire différents cercles dans des nuages de points entremêles ?

#### Choix de la modélisation du meilleur cercle passant par un nuage de points

Différentes façons de modéliser un cercle ont été publiées, il n'est pas utile d'entrer dans les détails, car il existe de nombreuses références qui traitent ce sujet [Gander *et al.*, 1994]. Un cercle étant défini par trois points, il existe évidemment différentes façons de faire passer

un cercle par un ensemble de plus de trois points, soit en minimisant la distance algébrique entre le cercle et les différents points, soit en minimisant la distance géométrique. On peut bien sur utiliser toute autre notion de distance pour le calcul. Mais celles pertinentes pour notre problème sont les distances algébrique et géométrique. Chacune a ses avantages et inconvénients qui sont maintenant rappelés.

*Minimisation de la distance algébrique*

Avantage de cette modélisation : il s'agit d'un système linéaire, avec une complexité de calcul réduite, et donc un temps de calcul faible. Inconvénient : cette minimisation n'a pas un sens physique évident, et en outre l'estimation des paramètres du cercle se fait d'une façon peu précise.

Le modèle mathématique employé pour la définition du cercle est le suivant :

$$F(x) = a\mathbf{x^T x} + \mathbf{b^T x} + c, o\ a \neq 0, \mathbf{x}\ et\ \mathbf{b} \in R^2 \tag{5.5}$$

Dans ce cas, pour ajuster un cercle à un nuage de points, il faut calculer $a$, $\mathbf{b}$ et $c$. En insérant les coordonnées des points dans l'équation 5.5 nous obtenons un système d'équations tel que :

$$B\mathbf{u} = 0, \tag{5.6}$$

où $u = \begin{bmatrix} a\ b_1\ b_2\ c \end{bmatrix}^T$ et $B = \begin{bmatrix} X_{11}^2 + X_{12}^2 & X_{11} & X_{12} & 1 \\ X_{21}^2 + X_{22}^2 & X_{21} & X_{22} & 1 \\ ..... & .... & .... & .... \\ X_{n1}^2 + X_{n2}^2 & X_{n1} & X_{n2} & 1 \end{bmatrix}$. On note que $X$ contient les coordonnées en $x$ et en $y$ de chaque point. En d'autres termes pour le point $H_i$, on a $X_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$.

Pour trouver la solution de l'équation homogène 5.6 nous imposons la contrainte $\|u\| = 1$, et nous cherchons donc à minimiser le système suivant :

$$\min_{\|u\|=1} \|Bu\| \tag{5.7}$$

Ce système sera résolu par une décomposition $SVD$ [1] et la solution finale sera celle de la plus petite valeur propre du vecteur propre correspondant. Les coordonnées du centre du cercle K (Figure 5.2) sont égales à :

$$K = (k_1, k_2) = \begin{bmatrix} \frac{-b_1}{2a} \\ \\ \frac{-b_2}{2a} \end{bmatrix} \tag{5.8}$$

Le rayon du cercle est calculé de la manière suivante :

$$r = \sqrt{\frac{\|b\|^2}{4a^2} - \frac{c}{a}}. \tag{5.9}$$

En outre, dans notre cas de figure le cercle passe par l'origine $O$, donc $c = 0$. Ainsi la seule inconnue du système est le centre du cercle.

1. SVD : Singular Value Decomposition

*Minimisation de la distance géométrique*

Une façon de définir un cercle est de minimiser une vraie distance, au sens géométrique. L'avantage de cette minimisation est qu'elle est plus précise que la précédente [Gander *et al.*, 1994]. En outre, elle possède un sens physique évident. Par contre, son inconvénient est qu'elle implique la résolution d'un système non linéaire, qui donc exige une solution approchée. Nous avons, pour trouver celle-ci, utilisé la minimisation algébrique décrite précédemment. En outre, il s'agit d'un calcul itératif qui entraîne un temps de calcul nettement plus élevé. Il s'agit alors de la minimisation de la somme des carrés des distances :

$$d_i^2 = (\left\|\mathbf{K} - \mathbf{H_i}\right\| - r)^2 ou\ r = \sqrt{k_1^2 + k_2^2}. \tag{5.10}$$

On rappelle que le cercle passe par l'origine. $\mathbf{K} = \begin{bmatrix} k_1 & k_2 \end{bmatrix}$ étant le centre du cercle, et $r$ son rayon. On considère $u = \begin{bmatrix} k_1 & k_2 \end{bmatrix}^T$ et nous cherchons à minimiser :

$$\min(\sum_{i=1}^{m} u^2.d_i) \tag{5.11}$$

### 5.3.4 Extraction des cercles dans plusieurs nuages de points entremêlés

Le cœur de l'implémentation algorithmique de la méthode et le bon fonctionnement de celle-ci repose entièrement sur l'efficacité de cette étape. Une méthode d'extraction des différents cercles a été inspirée de la méthode RanSac [Fischler et Bolles, 1981]. Certaines modifications ont du être apportées afin de pouvoir l'adapter à notre cas de figure. Les différentes étapes de l'algorithme sont les suivantes :

1. Choix de deux points $H_i$ au hasard dans l'ensemble des points $H_i$ disponibles, puisque pour obtenir un cercle passant par l'origine deux points suffisent.
2. Calcul du cercle passant par les points piochés et l'origine. Le modèle mathématique choisi ici est *la minimisation de la distance algébrique*.
3. Recherche de tous les autres points $H_i$ qui sont susceptibles de contribuer à ce cercle. Pour ce faire il est nécessaire d'avoir déterminé au préalable un seuil de capture, c'est à dire une bande autour de ce cercle : si un point $H_i$ est dans cette bande, il est retenu pour la définition du cercle, et sinon il est écarté. Le mode de calcul de ce seuil, à partir de la variance des points $H_i$, est explicité plus loin.
4. Identification du nombre de points sélectionnés, re-itération des étapes 1, 2, 3, et finalement conservation au cours de ces essais successifs de celui qui a capturé le plus de points. Cette opération est réitérée un grand nombre de fois afin de s'assurer que le nombre de points capturés ne peut pas être dépassé.
5. A l'issue de ce processus, on retient cet ensemble, et les points capturés correspondants sont alors retirés de l'ensemble de départ. Pour les points ainsi sélectionnés, on cherche alors le meilleur cercle par moindres carrés, mais cette fois ci à l'aide de *la minimisation géométrique*, avec pour valeur approchée les paramètres obtenus dans le calcul de la minimisation algébrique. Ce calcul est mené en prenant en compte les incertitudes liées à chaque point $H_i$. Cette étape fera l'objet de la section 5.4.

6. L'extraction des cercles est interrompue lorsqu'il ne reste qu'un tout petit nombre de points que l'on fixera arbitrairement, par exemple à 5, des points $H_i$ initialement détectés.

Le bon déroulement de l'extraction des cercles repose donc entièrement sur la bonne définition de la valeur $\epsilon$ de la largeur de la bande de capture. Par exemple si cette valeur est trop petite un nombre considérable de cercles sans sens géométrique réellement différents sera trouvé, et par contre si cette valeur est trop grande, on fera contribuer à un cercle donné, et donc au point de fuite correspondant, des points qui en réalité devraient être associés à un autre cercle. Il s'avère donc capital de pouvoir définir le seuil de capture de façon automatique sur une image donnée, et il ne serait sans doute pas très adapté d'utiliser le même seuil pour des images avec par exemple différentes résolutions. L'utilisation d'une méthode manuelle pour définir ce seuil causerait une grosse perte de temps et ferait perdre un intérêt majeur de la méthode. Dans l'optique d'une automatisation complète du processus nous avons donc cherché à extraire des paramètres indépendants du format de l'image, le paramètre $\epsilon$ devant de toute évidence être lié à la précision de la détermination des points $H_i$.

Dans notre étude, pour touver un seuil intrinsèque à l'image, nous avons pensé aux ellipses d'incertitudes.

En effet dans la Figure 5.8, il peut être vu que les ellipses d'erreur forment un bandeau autour des points $H_i$ et, expérimentalement, nous avons choisi la valeur médiane des grands axes des ellipses (à 39 % de confiance) comme un seuil satisfaisant pour cette capture utilisant la méthode RanSac [Fischler et Bolles, 1981]. On a donc ($i$ étant l'indice désignant chaque point $H_i$), $\epsilon$ étant la médiane des valeurs des demi-axes des ellipses d'erreur [Cooper, 1987] :

$$\epsilon = mediane_i \sqrt{\frac{1}{2}(\sigma^2_{H_{i,x}} + \sigma^2_{H_{i,y}} + \sqrt{(\sigma^2_{H_{i,x}} + \sigma^2_{H_{i,y}})^2 - 4\sigma^2_{H_{i,x}}\sigma^2_{H_{i,y}} - \sigma^2_{H_{i,x}H_{i,y}}})} \tag{5.12}$$

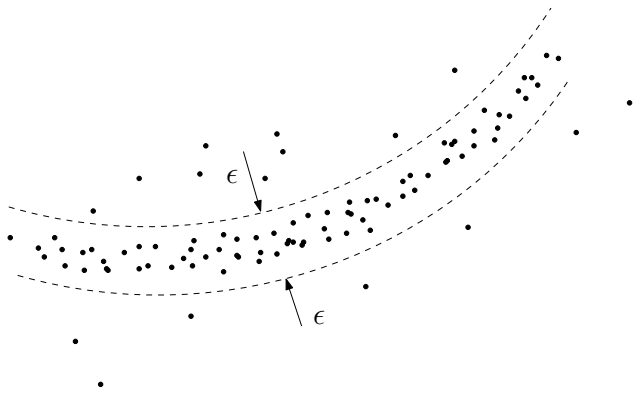
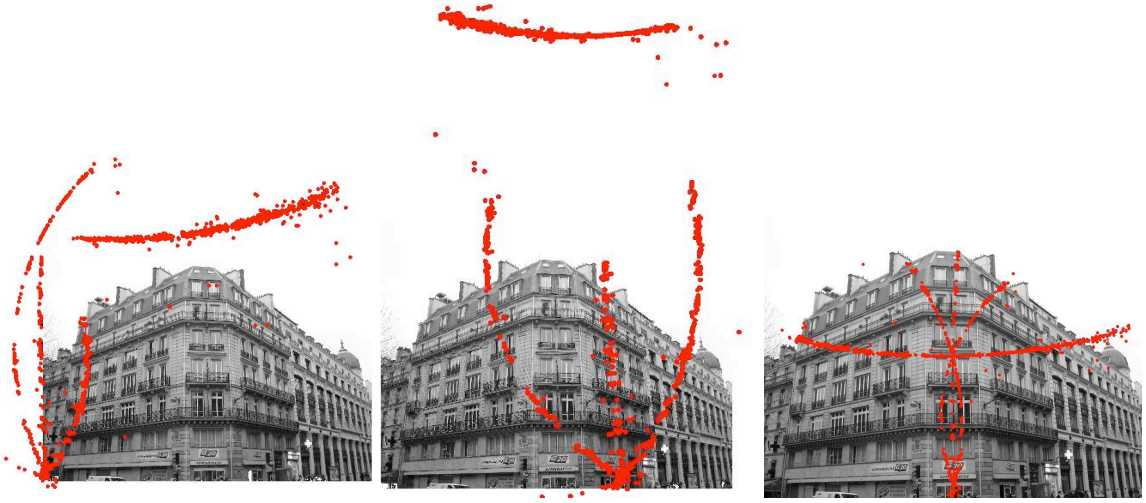



**Fig. 5.9.** Représentation de $\epsilon$ et de la bande de capture des points $H_i$ au sein du nuage de points.

### 5.3.5 Discussion sur le choix de l'origine

Un des points important à développer est le choix de l'origine. En effet dans les sections précédentes tous les points $H_i$ ont été définis par rapport à l'origine de l'image qui est en haut

à gauche. Le choix de l'origine a été discuté dans l'article de [Brauer Burchardt et Voss, 2000]. Mais dans cet article le cas d'un seul cercle a été traité. Le problème auquel nous sommes confrontés c'est de choisir une origine unique et optimale pour tous les cercles. Nous avons conçu un algorithme en 2 étapes, la première étape consistera à détecter les cercles, avec une origine en haut et à gauche de l'image. Dans la deuxième étape le barycentre des centres des cercles détectés est calculé. Ce barycentre sera ensuite utilisé comme étant l'origine idéale. L'avantage d'utiliser le barycentre comme origine est que la distance de tous les cercles à celle-ci est la même. Le choix de l'origine définira l'intersection des cercles entre eux, le but étant d'optimiser les intersections des cercles (angles d'intersection les plus grands possibles) afin que l'algorithme de détection de cercle fonctionne de manière satisfaisante. Par exemple il est déconseillé de prendre l'origine au milieu de l'image car les cercles crées se coupent plus d'une fois dans l'image, voir Figure 5.10.



(a) (b) (c)

**Fig. 5.10.** Exemple des différentes configurations des points $H_i$ en fonction de l'origine : a) l'origine est prise en haut à gauche de l'image, b) l'origine est sur le barycentre des cercles précedemment détectés, la géométrie des intersection des cercles est la plus favorable, c) configuration très défavorable des points $H_i$, l'origine est prise au centre de l'image.

## 5.4 Détermination des points de fuite et de leurs incertitudes

Une fois les paramètres du cercle obtenus, c'est à dire uniquement les coordonnés de son centre $(Xc, Yc)$ puisque le cercle passe par l'origine, les coordonnées du point de fuite $P$ sont calculés très facilement car le point $P$ se trouve sur le diamètre $OP$ (Figure 5.2). Ces coordonnées sont donc les suivantes :

$$X_P = 2k_1, \; Y_P = 2k_2 \tag{5.13}$$

La détection des segments dans les images, soit manuellement, soit comme cela été vu précédemment dans la section xxx à l'aide des opérateurs comme celui de Canny-Dériche,

n'est jamais fiable à 100 %. Ceci n'est pas bien grave pourvu qu'on sache modéliser ce taux de confiance. En outre, notre but est aussi de pouvoir calculer l'impact de l'imprécision des segments détectés sur la localisation du point de fuite. Celle-ci sera donc exprimée accompagnée de sa matrice d'incertitude. Dans cette partie nous démontrons comment à partir de l'incertitude sur le segment nous arrivons à modéliser son impact sur le point de fuite.

En supposant que la distribution des erreurs sur les segments suit une loi gaussienne, la méthode de Gauss-Helmert (GH) [Helmert, 1872] [Cooper, 1987] est employée. Quelques rappels sur celle-ci faciliteront la compréhension de nos travaux.

*La méthode de Gauss-Helmert*

La méthode GH est une généralisation des moindres carrés classiques. Sa grande différence avec la forme conventionnelle des moindres carrés est que les observations ont le "droit de bouger" de façon proportionnée à leur incertitude. En d'autres termes, elles sont modifiées à chaque itération, et donc ne sont pas constantes. D'autre part les inconnues et les observations sont entremêlées, ce qui ne permet pas d'utiliser des moindres carrés pondérés. La méthode GH va faire en sorte de modifier toutes les observations, afin de donner la meilleure solution, au sens des moindres carrés [Cooper, 1987].

En général, les éléments qui ont été mesurés sont dénotés par le vecteur $l$ de dimension ($m$ x 1), et le vecteur dont les éléments ne sont pas mesurés, mais estimés directement avec la méthode des moindres carrées, est noté par le vecteur $x$, de dimension ($u$ x $l$). La fonction de coût des relations entre les observations et les inconnues (au nombre de $c$), est exprimée de la manière suivante :

$$f(x, l) = 0, \tag{5.14}$$

où $f$ est le système des fonctions $f_i$.

Toutes observations dans la vie réelle incluent nécessairement une part de bruit. Nous dénotons par $\bar{l}$ la mesure supposée parfaite. La valeur supposée parfaite des inconnues est noté par $\overline{x}$. La fonction s'écrit donc :

$$f(\overline{x}, \bar{l}) = 0. \tag{5.15}$$

En général les $c$ fonctions $(f_i)$ sont non linéaires. Une linéarisation est effectuée, à l'aide du théorème de Taylor :

$$f(\overline{x}, \bar{l}) = f(x_0, l_0) + (\frac{\partial f}{\partial x})_0(\overline{x} - x_0) + (\frac{\partial f}{\partial l})_0(\bar{l} - l_0). \tag{5.16}$$

Les vecteurs $x_0$ et $l_0$ sont des approximations au premier ordre des valeurs réelles $\overline{x}$ et $\bar{l}$. Les coefficients des dérivées partielles sont évaluées, avec comme valeurs initiales $x = x_0$ et $l = l_0$. Si ($c$ x $u$ ) est la matrice jacobienne en fonction du vecteur des inconnues $x$, nous aurons :

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_u} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_u} \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial f_c}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_c}{\partial x_u} \end{bmatrix} = A \tag{5.17}$$

Et pour la matrice jacobienne fonction des observations, nous aurons donc :

$$\frac{\partial f}{\partial l} = \begin{bmatrix} \frac{\partial f_1}{\partial l_1} & \frac{\partial f_1}{\partial l_2} & \cdots & \frac{\partial f_1}{\partial l_m} \\ \frac{\partial f_2}{\partial l_1} & \frac{\partial f_2}{\partial l_2} & \cdots & \frac{\partial f_2}{\partial l_m} \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial f_c}{\partial l_1} & \frac{\partial f_1}{\partial l_2} & \cdots & \frac{\partial f_c}{\partial l_m} \end{bmatrix} = B \tag{5.18}$$

Nous avons donc :

$$A(\overline{x} - x_0) + B(\bar{l} - l_0) + f(x_0, l_0) = 0. \tag{5.19}$$

Les mesures $l$ peuvent être utilisées à la place de l'aproximation $l_0$, ainsi :

$$A(\overline{x} - x_0) + B(\bar{l} - l_0) + f(x_0, l) = 0. \tag{5.20}$$

Le vecteur $\bar{l} - l$ représente la différence entre la valeur réelle des observations et la valeur mesurée, en d'autres termes, la correction qu'il faut appliquer à $l$ pour obtenir $\bar{l}$. Cette correction $v$ est parfois nommée le vecteur de résidus. Si on nomme $\overline{x} - x_0$ par $x$ et $b = -f(x_0, l)$, la forme linéaire de l'équation 5.15 sera :

$$Ax + Bv = b. \tag{5.21}$$

La résolution de ce système se fait par la minimisation de :

$$\sum_i v_i^T W v_i, \tag{5.22}$$

$v$ étant le vecteur des résidus, et $W$la matrice de poids des observations.

La résolution du système utilise le multiplicateur de Lagrange de façon itérative. A chaque itération les vecteurs $x$ et $v$ sont calculés. La convergence est constatée lorsque les valeurs obtenues n'évoluent plus. Après compensation, le vecteur des inconnues $x$ est donné par :

$$\hat{x} = \left[A^T (BW^{-1}B^T)^{-1} A\right]^{-1} A^T (BW^{-1}B^T)^{-1} b. \tag{5.23}$$

Le vecteur des résidus $v$ :

$$\hat{v} = W^{-1}B^T(BW^{-1}B^T)^{-1} \times \{I - A[A^T(BW^{-1}B^T)^{-1}A]^{-1}A^T(BW^{-1}B^T)^{-1})\}b \tag{5.24}$$

La matrice variance covariance des paramètres inconnus estimés peut être calculée à partir de la relation suivante :

$$\Sigma_x = \left[A^T (BW^{-1}B^T)^{-1} A\right]^{-1} \tag{5.25}$$

Maintenant, si on revient à notre problème d'estimation de cercles à l'aide des points $H_i$, nous cherchons à calculer la matrice variance covariance du point de fuite en fonction de la matrice variance covariance des points $H_i$ (équation 5.4).

Les différentes étapes du calcul sont les suivantes :

1. Propagation de l'incertitude du segment sur les coordonnées du point $H_i$ correspondant. Ceci a été vu en détail dans la section 5.3.2.

2. Choix d'un modèle mathématique et définitions des observations et des inconnues. Le modèle mathématique choisi est *la minimisation de la distance géométrique*. Ici il est impossible de choisir le modèle algébrique, car cela n'aura pas de sens au sens des moindres carrés. La fonction de coût est la suivante :

$$f(X_C, X_H) = \left\|X_C - X_H\right\| - \left\|X_C\right\| = 0 \tag{5.26}$$

   $X_C$ est le vecteur des inconnues du modèle, il s'agit des coordonnées du centre du cercle. $X_H$ est le vecteur des points $H_i$, en d'autres termes le vecteur des observations. Dans l'équation 5.26, le modèle est défini pour des valeurs exactes des observations, évidemment fictives. Afin de faire une différenciation dans la suite les valeurs $\overline{X_C}$ et $\overline{X_H}$ seront définies comme les valeurs théoriques exactes. En d'autres termes nous aurons un système de $c$ équations, $c$ étant le nombre de point $H$.

3. Résolution à l'aide de la méthode de Gauss-Helmert. En nous référant à l'équation 5.21, nous avons besoin des matrices $A$ et $B$. $A$ est la matrice Jacobienne par rapport aux inconnues, qui sont les coordonnées du centre du cercle, $B$ est la matrice Jacobienne par rapport aux observations qui sont les coordonnées des points $H_i$. Or comme nous avons la matrice variance-covariance des points $H_i$, nous injectons ces informations dans l'équation 5.21. Pour cela, la matrice de poids des observations est créée, nommée $W$. Pour N observations, la matrice de poids a une dimension de $N \times 2N$. Si on suppose que les points $H$ sont statistiquement indépendants les uns des autres, la matrice aura la configuration ci-dessous :

$$W = \begin{bmatrix} \Sigma_{H_1}^{-1} & 0 & \dots & 0 \\ 0 & \Sigma_{H_2}^{-1} & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \Sigma_{H_n}^{-1} \end{bmatrix} \tag{5.27}$$

   $\Sigma_{H_i}$ est la matrice variance covariance des points $H_i$ fournis par le détecteur de segments, matrice qui a été précédemment calculée (équation 5.4). L'estimation de $x$ et $v$ de ce système a été décrite dans le paragraphe 5.4.

   L'incertitude sur le point de fuite est alors donnée par :

$$\Sigma_P = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \Sigma_{X_C} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \tag{5.28}$$

   où la matrice d'incertitude sur les coordonnées du cercle $\Sigma_{X_C}$ est calculée à partir de l'équation 5.25. L'équation 5.28 permet donc de calculer l'incertitude sur le point de fuite $P$.

## 5.5 Evaluation et performances

Les méthodes possibles d'évaluation des algorithmes d'extraction automatique des points de fuite d'une image ne sont pas évidentes. On peut certainement évaluer leur efficacité en termes de nombres de points de fuite trouvés, mais il n'existe pas de méthode pour trouver de façon neutre la valeur de référence : la valeur servant de comparaison ne peut être trouvée que par évaluation visuelle. En outre, trouver une valeur statistique probante ne va pas non plus de soi : il est assez simple de fausser les résultats, soit en effectuant les tests sur des images issues de géométries voisines (ce qui fausse le caractère aléatoire de l'échantillon), soit en choisissant des images trop simples (tous les résultats sont alors favorables), soit inhabituellement complexes, tels des bâtiments peu courants où toutes les lignes sont courbes (et les résultats seront anormalement mauvais). Et malheureusement à ce jour il n'existe pas de bases de données de référence pour l'évaluation des points de fuite. Dans le cas présent, l'évaluation a porté sur une centaine d'images relativement variées (voir exemple ci-dessous Figure 5.11, pour lesquelles le nombre de points de fuite trouvés (allant de 0 à 6 selon les images) a été encourageant.

**Fig. 5.11.** Exemple de la base d'images utilisées

Dans le but d'évaluer l'algorithme de détection des points de fuite, deux questions principales doivent être posées :

1. **Est-ce que l'algorithme est capable de détecter tous les points de fuite ?** L'algorithme a été testé sur 100 images différentes acquises en milieu urbain. Les résultats obtenus sont les suivants :

| | |
|---|---|
| Pourcentage de détections correctes du point de fuite correspondant aux verticales | 100% |
| Pourcentage de détections correctes des points de fuites correspondant aux horizontales | 92% |

**Tableau 5.1.** Résultats sur la base d'images urbaines. Les détections incorrectes correspondent à des sur-détections et à des sous-détections.

La bonne performance de la détermination du point de fuite correspondant aux verticales est due au fait qu'il est unique, qu'il rassemble souvent beaucoup de segments, et que le cercle correspondant à celui-ci est toujours bien isolé. Pour les cercles des points de fuite horizontaux, c'est plus compliqué, leurs nombres peuvent varier de 1 à 5 en fonction de la scène.

Une partie importante des quelques mauvaises détections identifiées est liée aux quelques segments de l'image portés par des droites passant très près de l'origine, et qui typiquement engendrent des points H qui donnent lieu spontanément à des mauvaises attributions, ce qui est un artefact normal lié au choix de l'origine. Un remède simple peut être trouvé, qui consiste à éliminer de la recherche automatique les segments de ce type. Néanmoins les présents résultats ont été obtenus sans procéder à ce filtrage.

Une autre évaluation a été faite, a contrario, sur 50 images ne contenant pas de points de fuite afin de vérifier si l'algorithme trouve des points de fuite ou pas. Les résultats de cette évaluation sont les suivants :

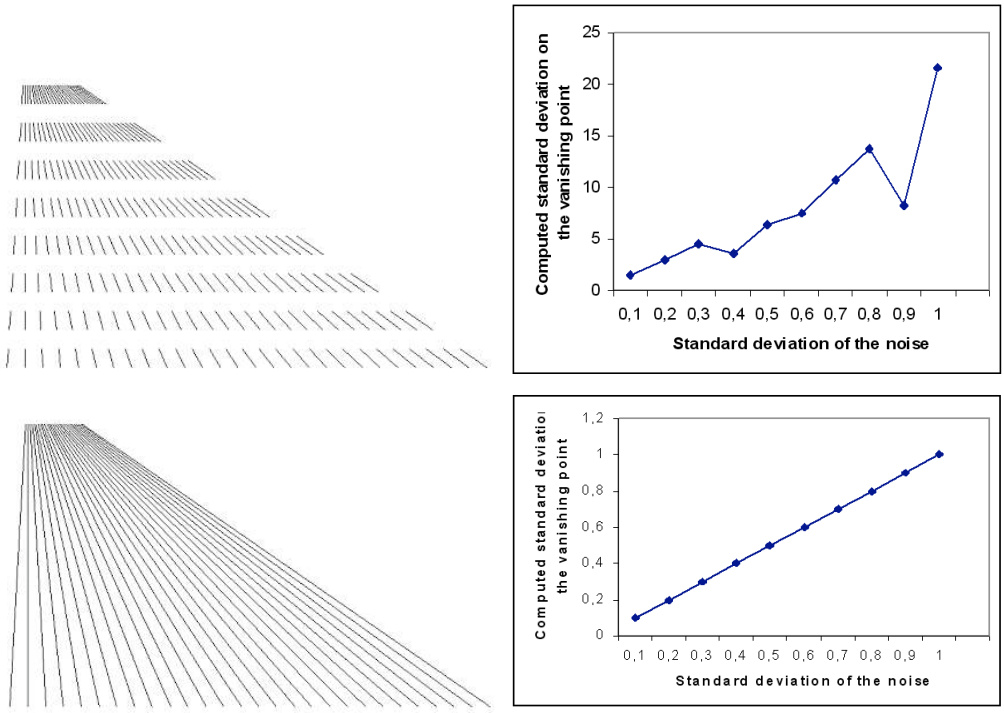
| | |
|---|---|
| Pourcentage de résultats non conformes | 4% |

**Tableau 5.2.** Résultats sur la base d'images ne contenant pas de points de fuite

2. **Est-ce que la propagation d'incertitude arrive à modéliser correctement la précision sur le point de fuite ?** Afin de pouvoir valider la propagation d'erreur sur le point de fuite, des images très simples comportant des points de fuite de coordonnées connues, avec des faisceaux de lignes droites se coupant de façon rigoureuse, ont été créées. Ces faisceaux de droites ont été traités ensuite en tiretés ou en traits d'axe afin de créer des segments de longueurs différentes.

   Ces images au format vecteur ont été ensuite transformées au format TIFF de façon à travailler sur des images de tailles différentes (Figure 5.12.a). Ensuite, toutes les étapes de l'algorithme ont été appliquées. Les extrémités des segments ont été bruitées avec un bruit gaussien variant entre 0.1 et 1 pixel, et la matrice variance covariance de chaque segment a été calculée, tous les éléments étant ainsi maîtrisés pour calculer la propagation d'erreur. Les résultats des graphes (Figure 5.12.b) montrent bien que la propagation arrive à retrouver correctement l'incertitude sur le point de fuite. Il est clair sur ces graphes que la longueur des segments a un impact direct sur la précision du point de fuite. La position du point de fuite influence aussi sa précision. Par exemple le point de fuite pour les verticales

aura, très logiquement, une meilleure précision en abscisse, et sera moins précis en $y$. En conclusion, on note qu'afin d'augmenter la précision sur le point de fuite, il faut avant tout améliorer la détection de segments.




(a) (b)

**Fig. 5.12.** a) Des ensembles de segments artificiels utilisés pour tester la détection des points de fuite. b) L'erreur sur la détermination du point de fuite en fonction du bruit apporté sur les extrémités des segments simulés.

A titre d'exemple, l'extraction des trois points de fuite et le calcul d'incertitude sur ceux-ci ont été effectués sur l'image de la Figure 5.13, les résultats sont les suivants (Table 2) :

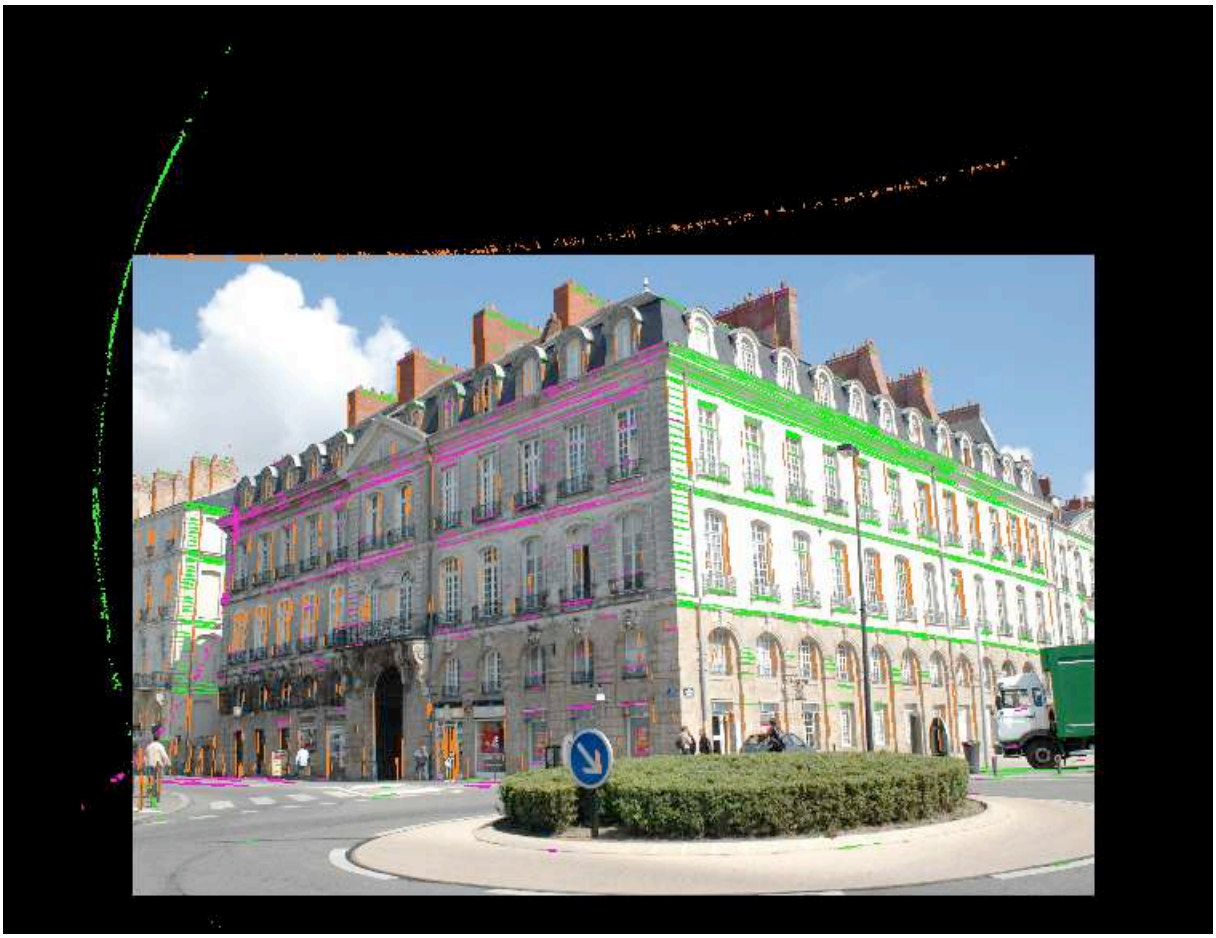


**Fig. 5.13.** Résultats de détection des points de fuite sur un bâtiment à Nantes

| Point de fuite | X (pixels) | Y (pixels) | $\sigma_x$ | $\sigma_y$ |
|---|---|---|---|---|
| Vertical (orange) | 1026 | -15788 | 7.2 | 117.6 |
| Horizontal 1 (vert) | 5148 | 1662 | 7.30 | 2.1 |
| Horizontal 2 (violet) | -1808 | 1652 | 2.8 | 0.5 |

**Tableau 5.3.** Les coordonnées des trois points de fuite extraits automatiquement de l'image de la Figure 5.13, avec leurs écart-types

### 5.5.1 Résultats en images

Quelques exemples de résultats de détection des points de fuite sur différentes images sont présentés (Figure 5.14).

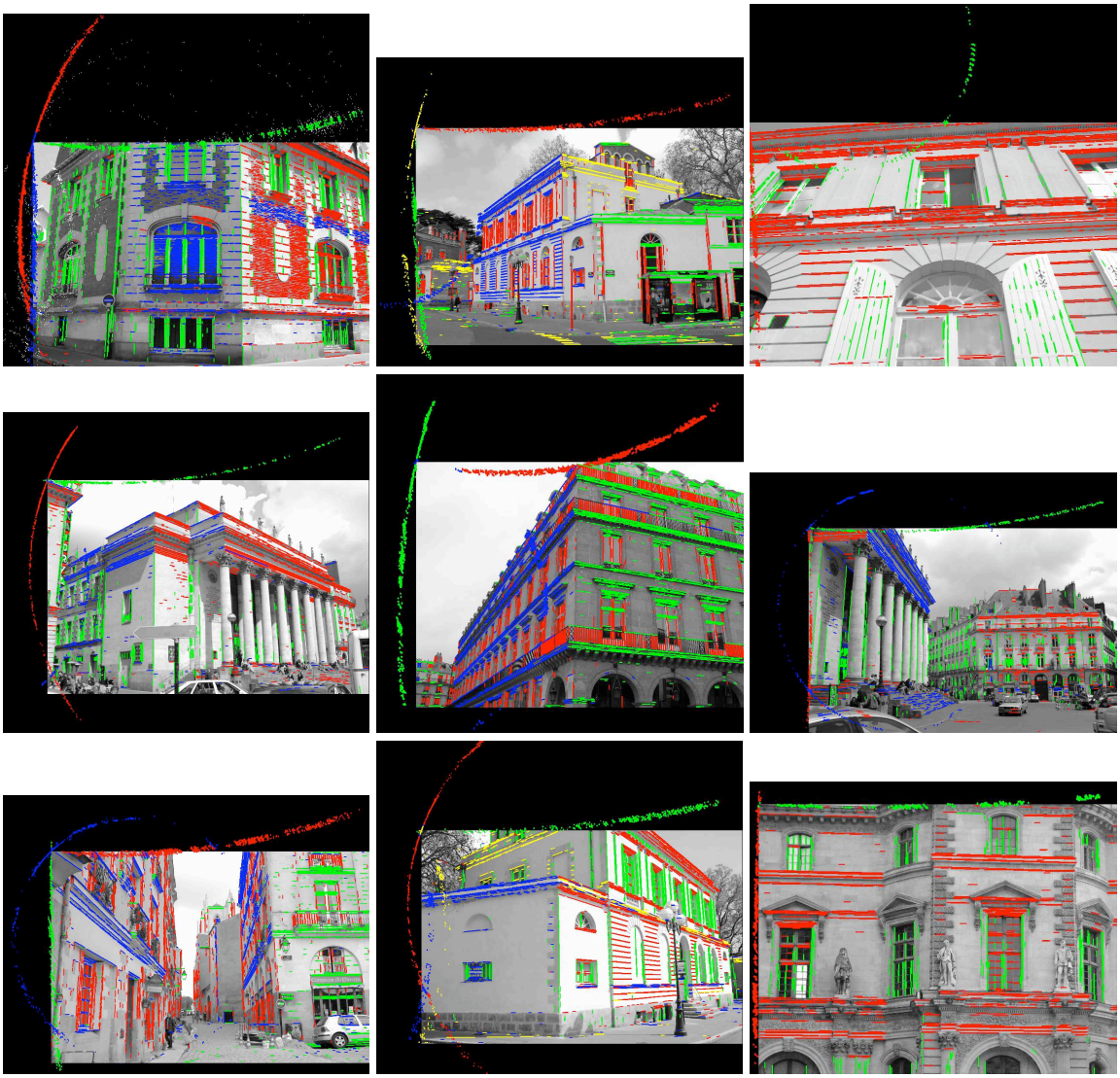


**Fig. 5.14.** Quelques résultats de détections : sur l'image originale, sont superposés les points H, dont les couleurs traduisent l'attribution automatique à tel ou tel cercle K (et donc tel ou tel point de fuite)

Avec l'aide de la détection des points de fuite, plusieurs points de fuite qui à l'œil ne sont pas détectés ont pu être extraits. C'est le cas des volets. En effet au premier abord, comme il peut être constaté sur la Figure 5.15, on peut dire que l'image ne comporte que deux points de fuite.

**Fig. 5.15.** Exemple d'un bâtiment contenant des volets.

Mais une fois l'algorithme appliqué, plusieurs points de fuite ont été détectés, à cause des volets sur les bâtiments. En effet ces volets forment un petit angle avec la façade. On peut voir ces résultats dans la Figure 5.16.

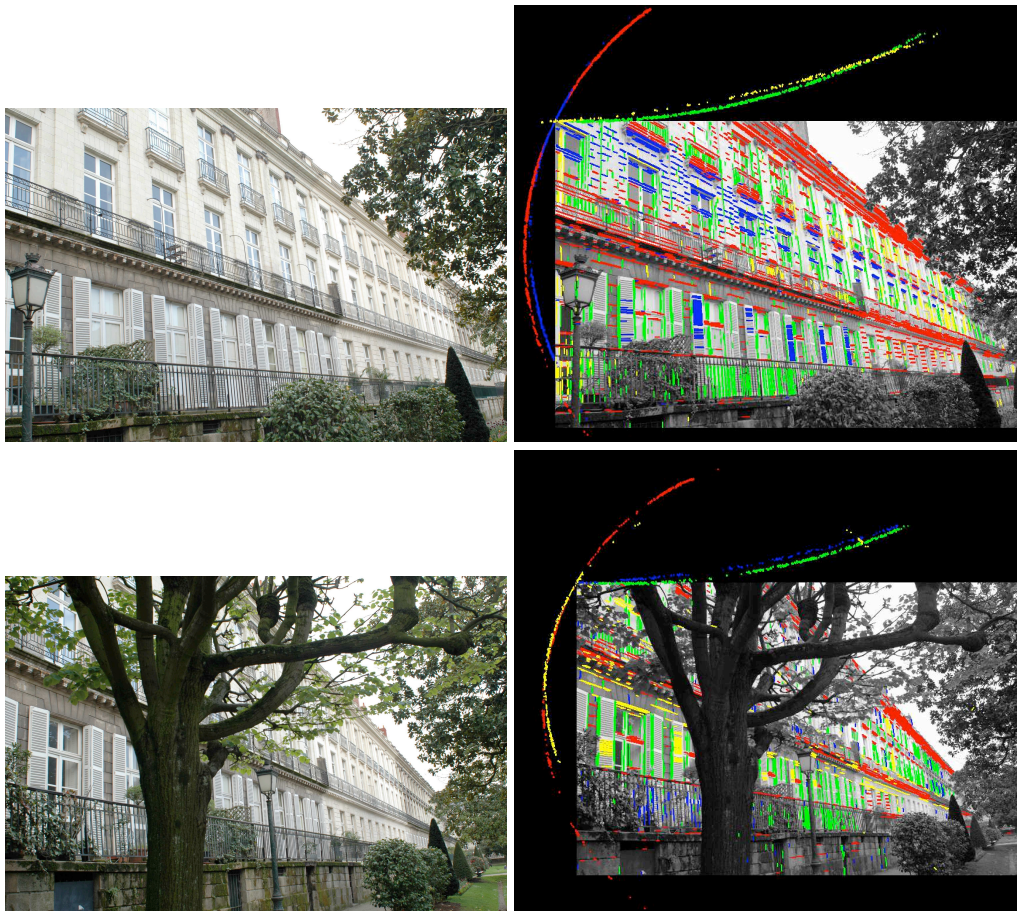


**Fig. 5.16.** Résultats de l'algorithme sur des bâtiments ayant des volets, montrant bien la sensibilité de la méthode.

### 5.5.2 Performances et temps de calcul

L'algorithme développé a donc une résolution angulaire très intéressante, puisqu'il est capable de séparer des segments appartenant à des plans d'orientation très proches (cas des volets).
La partie calcul des points de fuite, sans le calcul d'incertitude, se fait en temps réel (0.04 seconde avec un processeur intel Xeon, 1.60 Ghz et 2 Go de Ram, taille de l'image 3008 x 2000, 6 millions de pixels). Donc pour des applications temps réel, où dans un premier temps il n'est pas nécessaire d'avoir la précision sur le point de fuite, l'algorithme est opérationnel de manière satisfaisante. La partie calcul d'incertitude varie, avec cet ordinateur peu puissant et sans aucune optimisation spécifique, entre 30 secondes et 1 minute en fonction du nombre de segments de l'image.

## 5.6 Conclusion

Un algorithme entièrement automatique de détection de points de fuite dans des images de scènes urbaines a été présenté dans ce chapitre.
Cette approche travaille dans l'espace à 2 dimensions de l'image et s'appuie sur un théorème classique de géométrie projective (théorème de Chasles-Steiner, qui est une généralisation du théorème de Thalès pour l'ensemble des coniques), qui permet de transformer le problème de détection des points de fuite à partir de segments et leur incertitude en un problème de détection de cercles dans un nuage de points (chaque point correspond à un segment, et à chaque point on associe une incertitude). L'extraction de cercles utilise une méthode robuste de type RanSac [Fischler et Bolles, 1981], modifiée pour être très rapide par rapport à des techniques accumulatives (de type Hough ou autres).

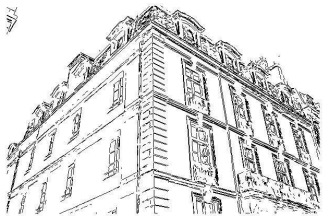
Notre apport principal a été la mise en place de la propagation d'incertitude à partir de la précision des segments détectés. Grâce à la connaissance des incertitudes il a ensuite été possible de définir, de manière intrinsèque à l'image, une tolérance de capture pour le RanSac. Ceci rend l'algorithme entièrement automatique, sans nécessité d'ajuster pour chaque image une tolérance.

Cette estimation robuste a été ensuite raffinée par une propagation d'incertitude par moindres carrés exploitant les variances individuelles de chaque segment. L'algorithme développé est robuste, sa précision est la meilleure au sens des moindres carrés compte tenu des incertitudes associées aux segments détectés, et en outre il est entièrement automatique.

Comme cela a été présenté dans les résultats, cet algorithme fonctionne aussi bien quelle soit la distance du point de fuite à l'image, même quand il va vers l'infini (voir chapitre 6.8), et il est extrêmement rapide.

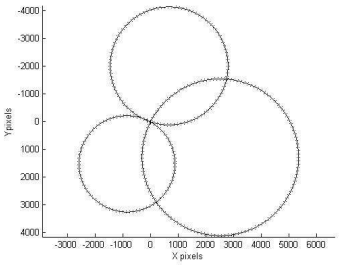
## 5.7 L'algorithme en quelques images

(étape 1)

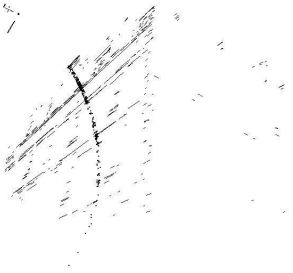


(étape 2)

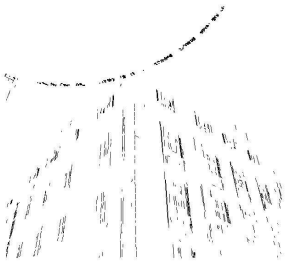
(étape 3)



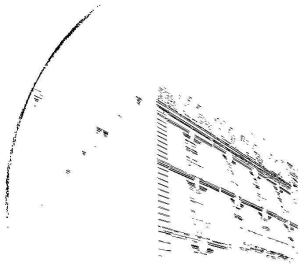

(étape 4)



(résultat 1)



(résultat 2)



(résultat 3)

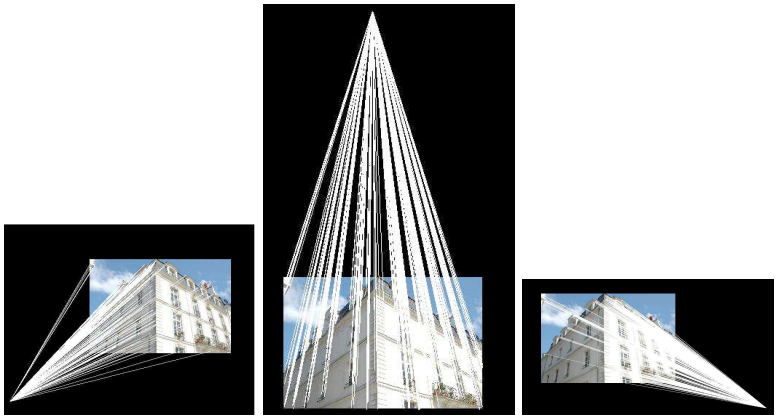


**Fig. 5.17.** Illustration des différentes étapes de l'algorithme

# 6

# Détection des points de fuite dans l'espace de la sphère de Gauss : méthode des plans

## Sommaire



## 6.1 Introduction

Introduit par Barnard [Barnard, 1983], le concept dit de la sphère de Gauss est celui d'une sphère de rayon d'unité avec pour origine le centre de perspective (le sommet de prise de vue). Chaque segment de l'espace image définit un plan avec le centre de perspective, généralement appelé plan d'interprétation. Les plans d'interprétation coupent la sphère de Gauss selon des grands cercles, et cette sphère est utilisée comme un espace d'accumulation pour ces grands cercles. Une des principales difficultés de cette approche est l'étape d'accumulation. En outre, la discrétisation engendre l'introduction d'un paramètre lié au pas de discrétisation qui n'est pas simple à gérer. Plusieurs de ces problèmes sont bien expliqués dans l'article de Shuffelt [Shufelt, 1999], qui propose une méthode d'extraction de points de fuite en faisant préalablement des hypothèses sur la géométrie de la scène.

[Collins et Weiss, 1990] ont proposé de travailler sur une sphère unité, ensuite ils utilisent des méthodes statistiques afin d'extraire les points de fuite. Reprenant ce concept de la sphère de Gauss, une nouvelle méthode de détection des points de fuite est proposée dans ce chapitre. Dans cette méthode l'extraction des points de fuite se fait sans l'étape d'accumulation, ce qui implique une vitesse bien plus élevée de l'algorithme, et ceci grâce à une complexité réduite. Le problème de détection des points de fuite est transformé en une recherche de plans coupant la sphère. Il est important de signaler que la méthode proposée fonctionne sans avoir aucune information préalable sur la scène et se fait de manière totalement automatique. D'autre part, elle peut s'affanchir de la connaissance des paramètres d'étalonnage de la caméra.

## 6.2 Présentation de la méthode

### 6.2.1 Définition de la géométrie du système

L'origine est choisie à une distance $D$ du plan image le long de la perpendiculaire à l'image issue de son centre $(Xo, Yo)$. Les axes $X$ et $Y$ sont parallèles aux axes de l'image (Figure 6.2.1).

A chaque segment de l'image est associé un plan passant par l'origine $O$ (cf Figure 6.2.1.a). Si l'origine $O$ est prise au centre optique, et donc à une distance focale $F$ de l'image, ce plan est classiquement nommé le plan d'interprétation [Barnard, 1983]. Mais, comme on le verra plus tard, la présente étude n'entraîne aucune obligation de prendre comme origine le centre optique, et $O$ peut être choisi librement hors du plan image. Notons que (cf. Figure 6.2.1.b) $L_1$ et $L_2$ sont les extrémités du segment dans l'espace objet, et $l_1$ et $l_2$ sont les projections de ceux-ci dans l'espace image.

Le plan passe donc par l'origine $O$ et la droite portant le segment (Figure 6.2.1).

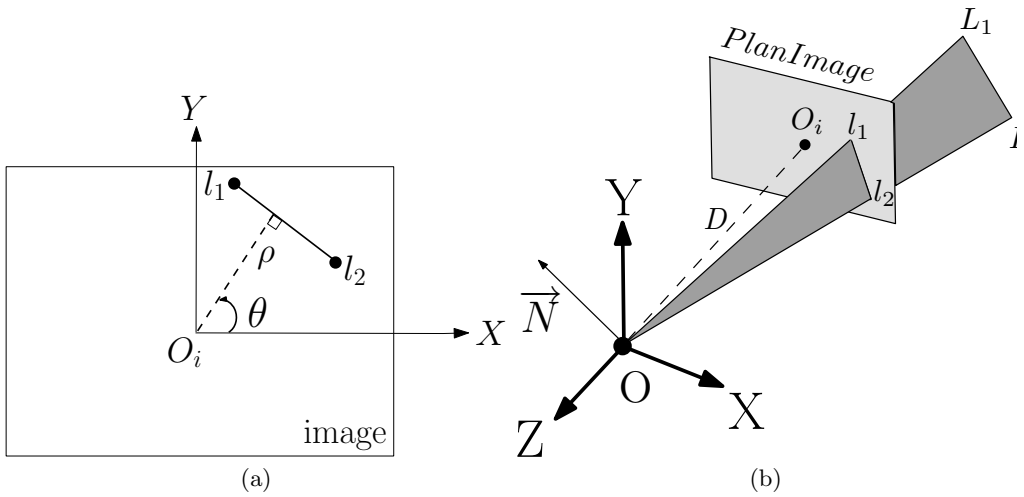



**Fig. 6.1.** a) Système d'axes de l'image. On rappelle l'équation polaire de la droite passant par le segment $l_1 - l_2$ : $X \sin\theta + Y \cos\theta = \rho$. b) Illustration du plan passant par l'origine $O$ et le segment $l_1$ $l_2$.

Le vecteur $N$ est la normale passant par $O$ au plan formé par $l_1$, $l_2$ et $O$. Le système d'axes utilisé ici a un $Y$ orienté dans le plan de l'image et vers le haut de celle-ci, et l'axe

des $Z$ est perpendiculaire au plan de l'image. Si chaque droite du plan $XY$ est définie par son équation polaire ($X \sin\theta + Y \cos\theta = \rho$) en fonction de $\rho$ et de $\theta$, coordonnées polaires de chaque segment dans le plan image comptées depuis l'origine de l'image (Figure 6.2), sa normale peut être définie de la façon suivante (équation 6.1) :

$$N_x = -\frac{D \sin\theta}{\sqrt{D^2+\rho^2}} \qquad N_y = -\frac{D \cos\theta}{\sqrt{D^2+\rho^2}} \qquad N_z = -\frac{\rho}{\sqrt{D^2+\rho^2}} \tag{6.1}$$

En normalisant tous les vecteurs normaux au plan issus de $O$, les extrémités de ceux-ci se retrouvent sur une sphère unité (Figure 6.2).

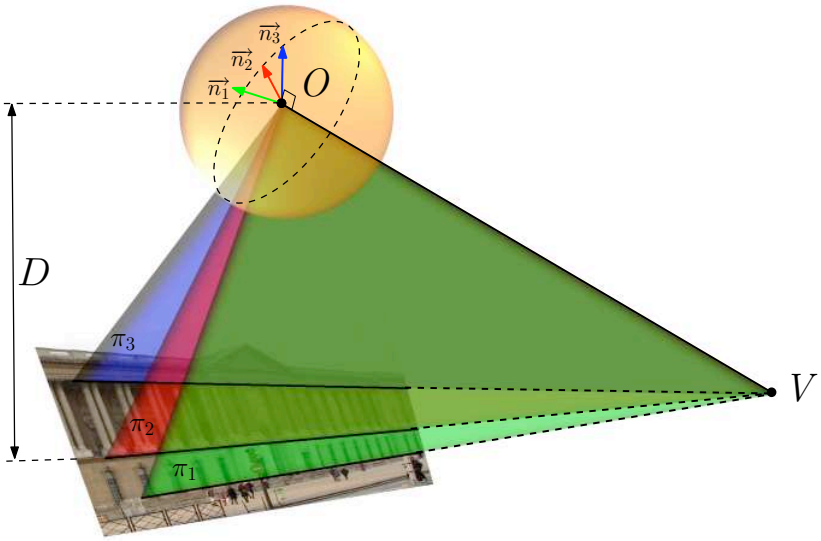


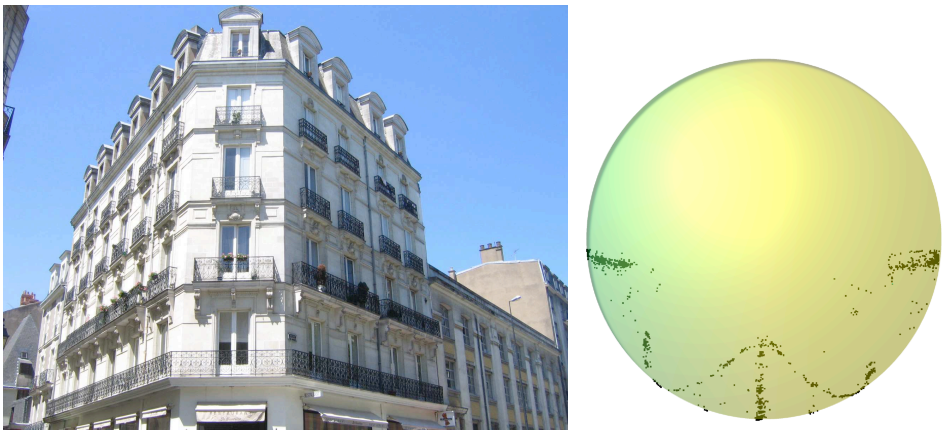

**Fig. 6.2.** Géométrie du système

Ainsi une "projection" sur une sphère unité a été effectué.

### 6.2.2 Géométrie et définition des plans de fuite

La méthode présentée ici est basée sur la propriété suivante des normales : *si les normales issues de $O$ correspondent à des segments d'une même gerbe perspective allant vers un point de fuite donné, elles sont coplanaires.*
Le plan défini par ces normales sera appelé par la suite *le plan de fuite*, car la normale issue de $O$ à ce plan perce le plan image sur le point de fuite $V$. La Figure 6.3 montre une image de bâtiment avec les normales correspondantes. Pour montrer les vecteurs normaux issus de $O$, leurs extrémités sont représentées sous forme de points.

Quelques auteurs ont utilisé cette géométrie : [Antone et Teller, 2000], [Weiss *et al.*, 1990]. Mais tous ont pris leurs origines sur le centre optique, donc à une distance F de l'image. Le problème traité ici est plus général, la valeur de $D$ comme la position du PPA pouvant très



(a) (b)

**Fig. 6.3.** a) Image d'un bâtiment qui contient quatre points de fuite, b) illustration des extrémités des normales sur la sphère unité.

bien être inconnus, faute d'étalonnage préalable. La méthode présentée ci-après n'exige en rien la connaissance de ces paramètres. La position de $O$ peut être prise de façon arbitraire, et pas nécessairement près du centre optique. En effet les familles de plans contenant le point $O$ ainsi que chaque segment passent nécessairement par le point de fuite relatif à celui-ci, où que soit situé ce point $O$ dans l'espace. Il est toutefois nécessaire de situer $O$ à une distance raisonnable de l'image si l'on ne veut pas rencontrer des problèmes purement numériques lors des calculs. En effet on voit bien que si $O$ est très distant du plan image, les plans de fuite vont devenir très proches d'un plan parallèle à l'image et seront donc très difficiles à séparer les uns des autres. Et inversement, si $O$ est très proche du plan image, les plans de fuite seront bien séparés mais ils seront très voisins d'une perpendiculaire au plan image, et donc leurs vecteurs normaux vont percer le plan image selon un angle très faible, ce qui sera donc aussi une source d'imprécision numérique. Les valeurs satisfaisantes pour $D$ peuvent varier assez largement, mais lorsque $D$ prend pour valeur la plus grande dimension de l'image, aucun problème d'imprécision numérique n'a été observé.

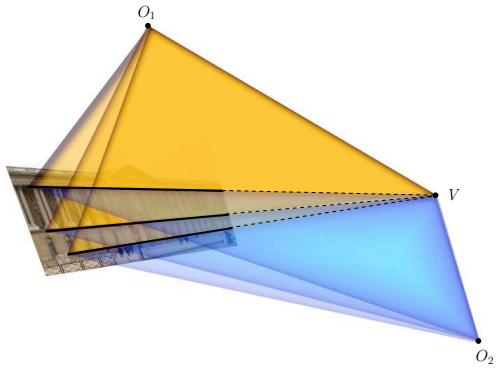



**Fig. 6.4.** Illustration des élements géométriques utilisés dans les algorithmes : l'ensemble des plans construits à partir des segments de l'image et contenant O se coupent en V, quelle que soit la position de O.

Le problème de détection des différentes familles de directions de l'image correspondant à des directions parallèles de l'espace objet revient donc à détecter tous les plans significatifs formés par les extrémités des normales issues de $O$ et de longueur unité. Dans la suite de ce chapitre la méthode d'extraction des plans sera approfondie et sera complétée par une étude sur la propagation d'erreur sur les résultats obtenus pour le point de fuite.

## 6.3 Algorithme de détection des points de fuite

L'algorithme proposé se déroule en 3 étapes :

1. détection des segments sur l'image,
2. extraction des plans de fuite,
3. propagation d'erreurs sur le point de fuite.

Nous ne détaillerons pas ici la partie détection de segments. Pour plus de détails, voir les chapitres 2.3.2 et 5.3.1.

### 6.3.1 Extraction des plans de fuite

Une fois les segments détectés sur l'image, ainsi que les droites portant ces segments, les normales peuvent être facilement calculées. Maintenant le problème posé est l'extraction des différents plans dans un ensemble de nuages de points. Pour cela une méthode inspirée du RanSac [Fischler et Bolles, 1981] a été implémentée. RanSac est un estimateur robuste qui repose sur le principe de tirage aléatoire. Tous les paramètres nécessaires pour le déroulement du RanSac sont brièvement décrits dans l'annexe A, sauf qu'ici il faut trouver le modèle adapté pour des nuages de points entremêlés. Dans le cas des plans de fuite, plusieurs nuages de points co-existent. Il a donc fallu apporter quelques changements à la méthode du RanSac. Les principaux portent sur :

- la tolérance sur l'erreur adaptée au modèle $t$,
- le nombre de tirages $N$,
- le nombre acceptable de données $T$ pour obtenir un consensus.

Le principe de l'algorithme est le suivant :

- 2 normales sont tirées de manière aléatoire $(N_1, N_2)$,
- calcul du produit vectoriel entre les 2 normales, le produit vectoriel de $N_1$ avec $N_2$ définit la normale au plan $(V_p)$,
- toute la pile des normales est parcourue pour trouver d'autres normales coplanaires avec les deux premières, avec la condition que $V_p$ leur soit orthogonale avec une tolérance $t$.
- Après $Nb$ itérations, le plan qui a agrégé le plus grand nombre de normales est gardé comme un plan de fuite,
- les normales classées dans un plan sont retirées de la pile principale,
- tout le processus précédent se répète jusqu'à ce qu'il n'y ait plus que 4 normales dans la pile.

Un des paramètres très importants qui a été étudié afin de pouvoir assurer une détection automatique des points de fuite est le calcul de la définition de la valeur de tolérance $t$. Là est l'essentiel de la différence entre divers algorithmes possibles, plus ou moins automatiques.

Cette valeur doit s'adapter aux caractéristiques de l'image. Ici la valeur de la tolérance est définie en fonction des matrices d'incertitude sur les paramètres de la droite sur l'image. Cette incertitude est ensuite propagée sur les normales aux plans contenant les segments et passant par $O$, et il peut en être déduit directement l'incertitude angulaire de la normale aux plans de fuite.

La médiane de toutes ces valeurs angulaires définira la tolérance $t$ (voir Figure 6.5 et l'algorithme 1).

Le nombre de tirages ne peut pas se calculer conventionnellement avec la méthode probabiliste donnée par Fichler et Bolles [Fischler et Bolles, 1981] car aucune connaissance à priori n'est disponible. Il est impossible de dire d'avance quel pourcentage de points est bon et quel pourcentage ne l'est pas. Une normale appartenant à un plan est un *inlier* pour son plan à lui et sera considérée comme un *outlier* pour un autre plan, il est donc impossible de trancher sur une stricte base probabiliste. De façon empirique, il a été trouvé satisfaisant de prendre pour $Nb$ la moitié du nombre de segments de l'image. Ainsi le choix de $Nb$ s'adapte tout seul à l'image, et donc le traitement reste entièrement automatique.

**Algorithme 1** Calcul de la tolérance $t$ pour le processus du RanSac

$Pile_{\alpha_i} =$
**for** $i = 0$ to *nombre de normales* **do**
  *Calcul de la matrice variance-covraiance des normales N en fontion de la matrice variance-covariance de* $\theta$ *et* $\rho$
  $\sum_{N_{x,y,z}} = J \sum_{\theta,\rho} J^t$ {J est la matrice jacobienne de la normale $N$ (équation 6.1) en prenant en compte de $\theta$ et $\rho$}
  $\overrightarrow{\delta N} = \overrightarrow{N} + (\sigma_{N_x}, \sigma_{N_y}, \sigma_{N_z})$
  $\alpha_i = |\arccos\left(\frac{(N|\delta N)}{||N|| \cdot ||\delta N||}\right)|$

  $Pile_{\alpha_i} \Leftarrow \alpha_i$
**end for**
*Calcul de la tolérance* $(t)$
**return** $t \Leftarrow$ *mediane de* $Pile_{\alpha_i}$

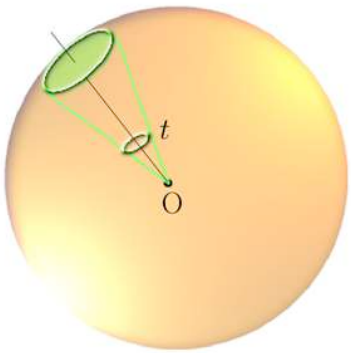
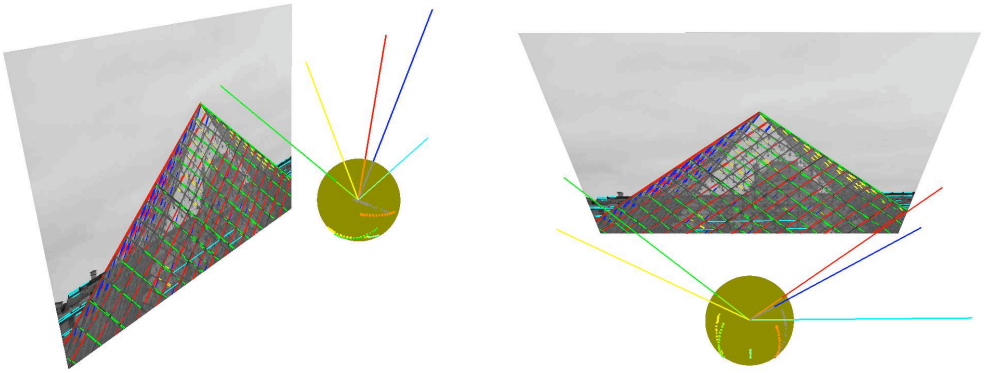



**Fig. 6.5.** Illustration de la tolérance $t$.

Un exemple de la détection des points de fuite avec l'algorithme des plans sur la pyramide du Louvre est proposé (Figure 6.6), ainsi qu'un autre résultat sur un bâtiment parisien contenant 4 points de fuite (Figure 6.7).



(a) (b)

**Fig. 6.6.** Illustration de la détection des points de fuite avec la méthode des plans.

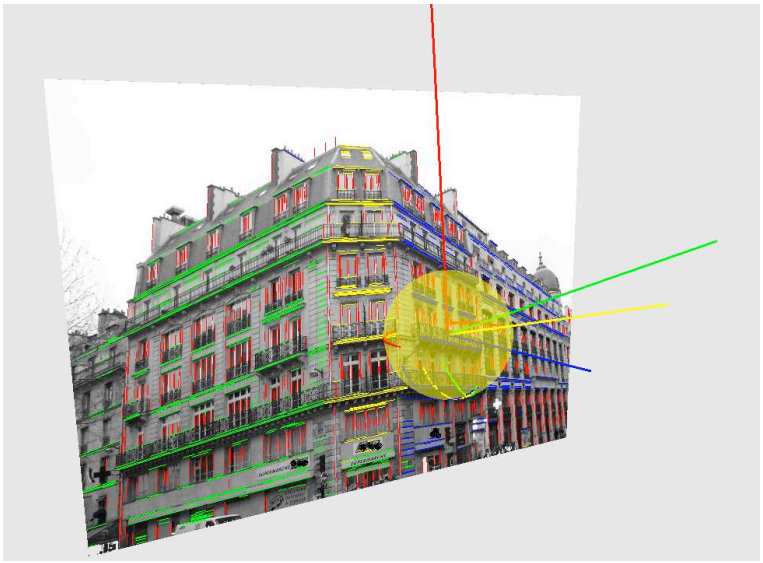


**Fig. 6.7.** Illustration de la détection des points de fuite avec la méthode des plans.

### 6.3.2 Propagation de l'erreur et analyse de l'incertitude sur le vecteur du point de fuite

Pour compléter la démarche de détection de points de fuite, le calcul d'incertitude sur la localisation du vecteur normal au plan de fuite est effectué. La démarche est similaire à celle présentée dans le chapitre 5.4 avec l'aide de la méthode de Gauss-Helmert. Notre objectif est de propager l'incertitude des segments sur la localisation du point de fuite. Pour cela dans un premier temps nous avons besoin de la propager sur celle des vecteurs normaux $N$.

La matrice variance-covariance du vecteur $N$ peut être aisément calculée directement en fonction de la matrice d'incertitude de son segment détecté c'est à dire en fonction de $\theta$ et $\rho$. En reprenant les relations de 6.1, et en appliquant la propagation d'erreur nous obtenons :

$$\Sigma_{N_i} = \begin{bmatrix} -\frac{(D)\cos(\theta)}{\sqrt{D^2+\rho^2}} & \frac{(D)\sin(\theta)\rho}{(D^2+\rho^2)^{3/2}} \\ \frac{(D)\sin(\theta)}{\sqrt{D^2+\rho^2}} & \frac{(D)\cos(\theta)\rho}{(D^2+\rho^2)^{3/2}} \\ 0 & -\frac{1}{\sqrt{D^2+\rho^2}} + \frac{\rho^2}{(D^2+\rho^2)^{3/2}} \end{bmatrix} \begin{bmatrix} \sigma^2_{i(\theta)} & \sigma_{i(\theta,\rho)} \\ \sigma_{i(\theta,\rho)} & \sigma^2_{i(\rho)} \end{bmatrix} \begin{bmatrix} -\frac{(D)\cos(\theta)}{\sqrt{D^2+\rho^2}} & \frac{(D)\sin(\theta)\rho}{(D^2+\rho^2)^{3/2}} \\ \frac{(D)\sin(\theta)}{\sqrt{D^2+\rho^2}} & \frac{(D)\cos(\theta)\rho}{(D^2+\rho^2)^{3/2}} \\ 0 & -\frac{1}{\sqrt{D^2+\rho^2}} + \frac{\rho^2}{(D^2+\rho^2)^{3/2}} \end{bmatrix}^T \tag{6.2}$$

### 6.3.3 Choix du modèle

L'équation d'un plan $P$ passant par l'origine et par un point du vecteur normal quelconque $N = [Nx, Ny, Nz]^T$, est définie de la manière suivante :

$$P \equiv aN_x + bN_y + cN_z = 0 \tag{6.3}$$

avec $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ comme normale au plan.

La distance de l'extrémité de la normale $N$ à ce plan est égale à :

$$d_{N,P} = \frac{|aN_x + bN_y + cN_z|}{\sqrt{a^2+b^2+c^2}} \tag{6.4}$$

En imposant que $\sqrt{a^2+b^2+c^2} = 1$ le modèle de distance est simplifié :

$$d_{N,P} = |aN_x + bN_y + cN_z| \tag{6.5}$$

Notre modèle sera alors :

$$d^2_{N,P} = (aN_x + bN_y + cN_z)^2 \tag{6.6}$$

En employant le modèle de Gauss-Helmert défini dans le chapitre 5.4, $Ax + Bv = b$. $A$ est la matrice Jacobienne par rapport aux inconnues (le vecteur normal du plan), $B$ est la matrice Jacobienne par rapport aux observations qui sont les $N$. $x$ est le vecteur des inconnues $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$, $v$ est le vecteur des observations. Par ailleurs nous avons aussi calculé la matrice variance-covariance pour chaque $N$, on peut alors utiliser ses matrices d'incertitude (équation 6.2). A ce système d'équations nous rajoutons la conditions de normalité $a^2+b^2+c^2 = 1$ car comme le plan passe par l'origine, il n'y a en fait que 2 inconnues indépendantes. La résolution de ce système se fait classiquement avec la méthode de Gauss-Helmert [Cooper, 1987].

## 6.4 Comparaison avec l'aide d'une méthode externe : utilisation d'un photothéodolite

On peut essayer de mesurer de façon indépendante et plus précise les coordonnées que l'on devrait trouver pour les points de fuite, et alors deux options sont envisageables : travailler sur des images de synthèse, ou employer un photo-théodolite. Celui-ci permet l'acquisition d'images avec une caméra étalonnée, et l'orientation de l'axe optique est mesurée avec une extrême précision grâce au théodolite qui est inclus dans la monture de la caméra. Pour plus d'information sur le photo-théodolite voir [Grenier, 2006].

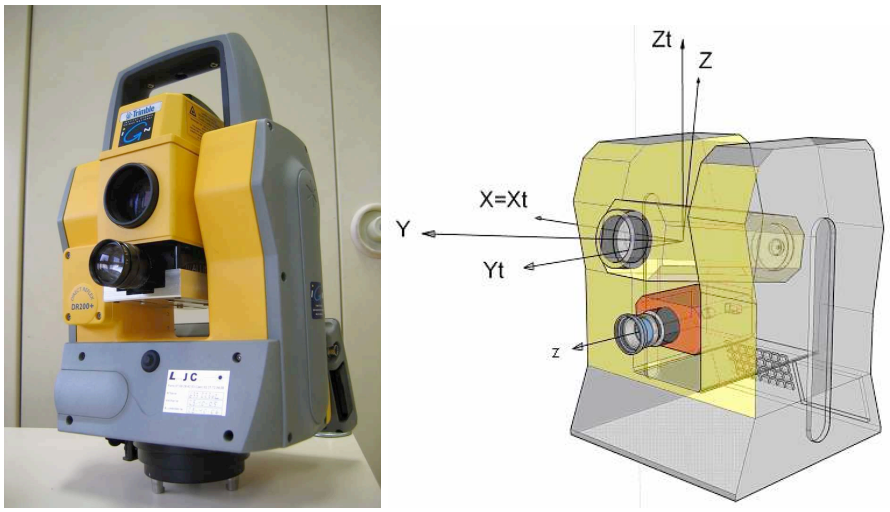



**Fig. 6.8.** Le photo-théodolite de l'IGN : il associe un tachéomètre Trimble 5603 à une caméra numérique MARLIN (Allied Vision Technologies). La caméra est située sous la lunette. Le repère lunette $(X, Y, Z)$ sera le plus utilisé : l'axe $Y$ est confondu avec l'axe de visée du tachéomètre (image et légende empruntées à Lazare Grenier [Grenier, 2006].

La validation a été menée de la façon suivante : trois images successives d'une même scène ont été prises, présentant un bâtiment dont les trois points de fuite correspondent à des directions orthogonales, avec des angles de prises de vue assez différents. Les angles du théodolite ont été enregistrés à chaque fois.

Les trois images ont été traitées afin d'en extraire de façon automatique les points de fuite. Les angles $\omega$ et $\phi$, calculés à partir de [Patias et Petsa, 1993] (voir chapitre 4.4) en utilisant les valeurs de distance focale et de position du PPA issues d'une calibration, sont directement liés aux angles $H$ et $V$ mesurés sur le théodolite. Les écarts d'orientation entre les images 1 et 2, puis 2 et 3 ont été comparés selon que l'on exploite les points de fuite ou les valeurs mesurées au théodolite : ils sont de l'ordre de 1 grade. Ceci correspond sur un segment de 1 m (longueur moyenne des segments de ces images) à une erreur de l'ordre de 1 cm, ce qui est donc un ordre de grandeur satisfaisant.

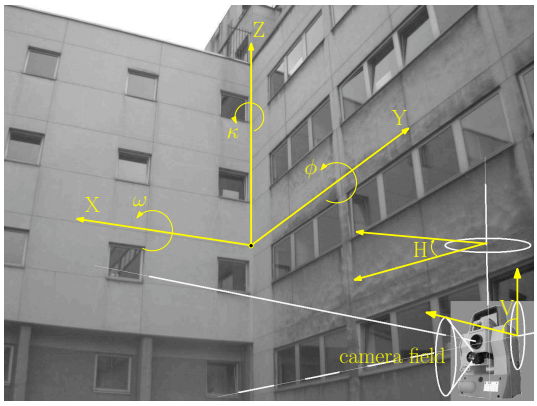



**Fig. 6.9.** Scène imagée pour la validation utilisant le photothéodolite. Illustration du système d'axes.

| Angles en grades | $\omega$ | $V$ | $\phi$ | $H$ |
|---|---|---|---|---|
| Im. 1 | 25.362 | 326.058 | 28.509 | 393.131 |
| Im. 2 | 33.809 | 333.601 | 46.415 | 411.355 |
| Im. 3 | 24.538 | 325.087 | 33.134 | 395.983 |

**Tableau 6.1.** Observations faites sur trois images à l'aide d'un photothéodolite

| Var Image | Var $\omega$ | $VarV$ | Dif | $Var\phi$ | $VarH$ | Dif |
|---|---|---|---|---|---|---|
| Im. 2/1 | −8.446 | −7.543 | −0.903 | −17.905 | −18.224 | 0.318 |
| Im. 3/2 | 9.270 | 8.514 | 0.756 | 13.280 | 15.372 | −2.091 |

**Tableau 6.2.** Mesure des changements d'orientation entre trois images successives d'une même scène en exploitant les points de fuite (angles $\omega$ et $\phi$) ou les mesures $H$ et $V$ du théodolite. Les différences (colonnes "Dif") sont petites, de l'ordre de 1 grade

## 6.5 Evaluation et performances

Suivant la même démarche adoptée dans le chapitre 5.5, et en travaillant sur la même base d'images urbaines (Figure 5.11) nous avons évalué l'efficacité de l'algorithme présenté pour la détection des points de fuite. Les résultats obtenus sont les suivants :

| | |
|---|---|
| Pourcentage de détections correctes du point de fuite correspondant aux verticales | 100% |
| Pourcentage de détections correctes des points de fuites correspondant aux horizontales | 96% |

**Tableau 6.3.** Résultats sur la base d'images urbaines (cf. Figure 5.11)

Comme on peut le noter, les résultats de détection des points de fuites des horizontales sont améliorés de 4% par rapport à la méthode des cercles. Ce qui démontre qu'en termes de résolution géométrique l'algorithme des plans est un peu plus performant que celui des cercles.

## 6.6 Temps de calcul

La partie calcul des points de fuite sans le calcul d'incertitude se fait en temps réel (0.02 seconde avec un processeur intel Xeon, 1.60Ghz et 2 Go de Ram, taille de l'image 3008 x 2000, 6 millions de pixels). Donc pour des applications temps réel, où dans un premier temps il n'est pas nécessaire d'avoir la précision sur le point de fuite, l'algorithme est opérationnel de manière satisfaisante.

## 6.7 Quelques résultats en images

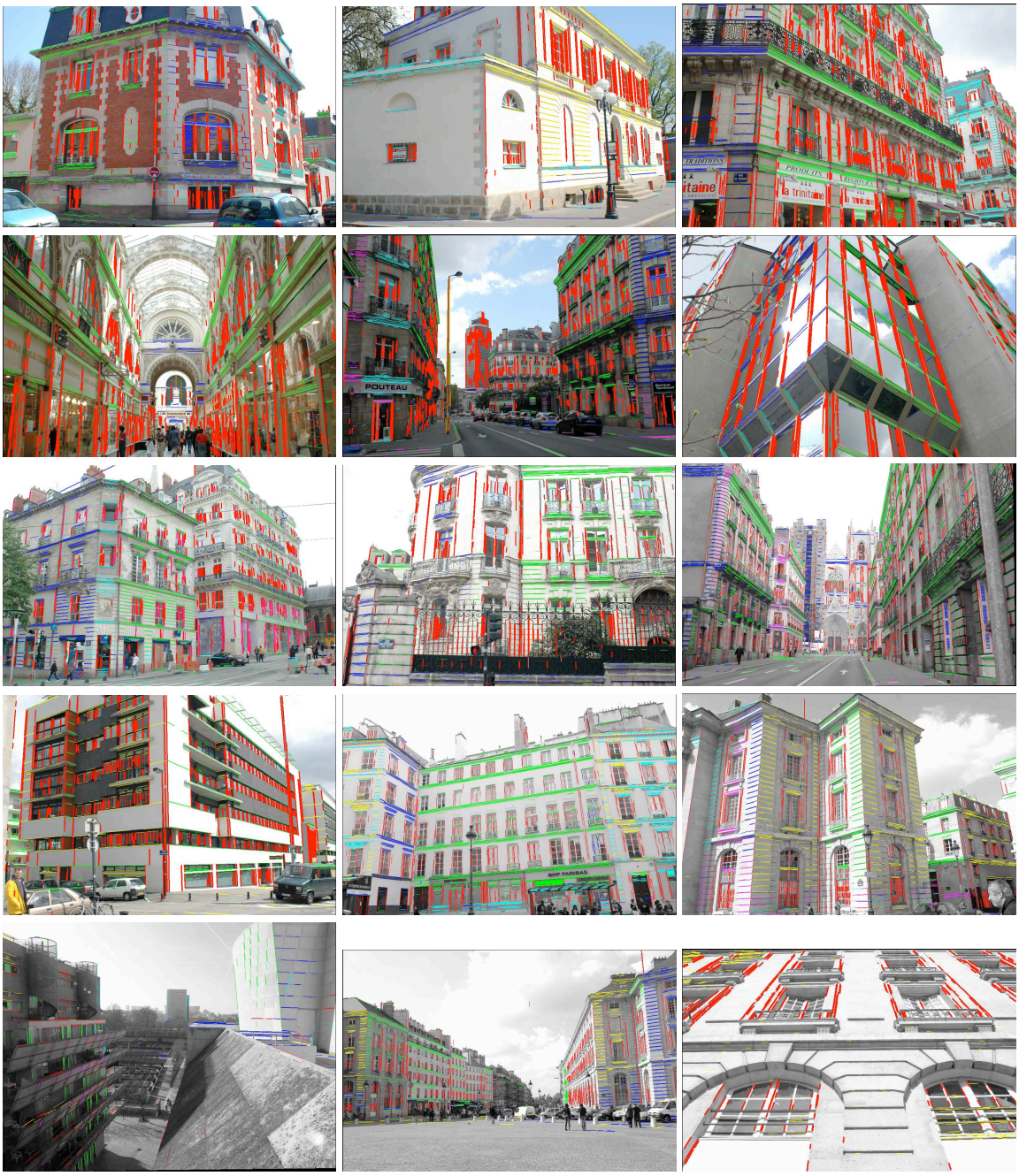



**Fig. 6.10.** Quelques résultats de détections avec la méthode des plans

## 6.8 Eléments de comparaison de la précision des algorithmes de détection de point de fuite

*Estimation de l'impact de la distance entre le point de fuite et l'image*

Trois algorithmes de détection de points de fuite ont été comparés : 1 - la méthode des cercles (chapitre 5), 2 - la méthode des plans (chapitre 6), 3 - la méthode plus classique basée sur l'intersection des segments ([Rother, 2002]). Les estimations de performances ont été faites avec l'aide de segments synthétisés, afin d'évaluer l'efficacité même quand le point de fuite est extrêmement loin de l'image. La dimension de l'image choisie est de 3008 x 2000 pixels, et la distance focale est de 3000 pixels. Pour chaque image, approximativement 100 segments, d'environ 200 pixels de long, sont générés. Les estimations utilisent deux niveaux de bruit gaussien, l'un avec un écart-type, jugé comme réaliste, de 0.2 pixel (pour chacun des deux points d'extrémités de chaque segment), et l'autre avec un écart type de 1 pixel, pour évaluer l'impact sur des situations avec un bruit élevé. Rappelons en effet (selon C.Thom, dans [Kasser et Egels, 2001],§ 1.9.3.1) : "En ce qui concerne les détails ponctuels, la précision est là toute trouvée : il n'y a aucun moyen de déterminer la position du détail à l'intérieur de son pixel. D'où une précision e.m.q. en x et y de 0,29 pixel, indépendante de la précision radiométrique".

Le système de référence est le même que dans la Figure 5.6, avec l'origine du système dans le coin supérieur gauche de l'image. Les coordonnées $x$ du point de fuite sont fixées à 1504 pixels. Pour chaque nouvelle itération, 1000 pixels sont enlevés à la coordonnée $y$ du point de fuite artificiel qui varie donc de 0 à $-1000000$ pixels. On peut considérer que pour une image de 3008 x 2000 pixels, 1000000 pixels est une valeur quasiment infinie (Figure 6.11). L'erreur angulaire est définie comme la différence des angles entre la direction théorique du point de fuite et la direction du point de fuite estimé.

Comme on peut le voir dans les Figures 6.12 et 6.13, le résultat important est que la distance plus ou moins grande du point de fuite n'a pas du tout d'impact sur la précision de détermination de sa direction, pour les nouvelles méthodes utilisées (1 et 2 ci-dessus). En d'autres termes, la méthode des cercles, comme la méthode qui utilise la sphère de Gauss (on peut noter que cette dernière est nettement moins sensible au bruit des segments), opèrent de façon satisfaisante quelle que soit la localisation du point de fuite. Le facteur majeur qui influence la détermination de la localisation du point de fuite est, comme prévisible, le bruit sur les segments. En l'absence de bruit, l'erreur angulaire est toujours nulle, montrant la stabilité de la méthode de calcul. On note sans surprise que pour un bruit de 0.2 pixel, l'erreur trouvée pour la direction du point de fuite est approximativement 5 fois plus faible que pour un bruit de 1 pixel. Par contre la méthode basée sur l'intersection des segments est fortement influencée par la distance du point de fuite à l'image. Même dans un cas de faible bruit l'erreur angulaire dépasse souvent 5 degrés 6.14. En conclusion on peut dire que les méthodes basées sur l'intersection des segments ne sont pas du tout adaptées aux points de fuite qui tendent vers l'infini.

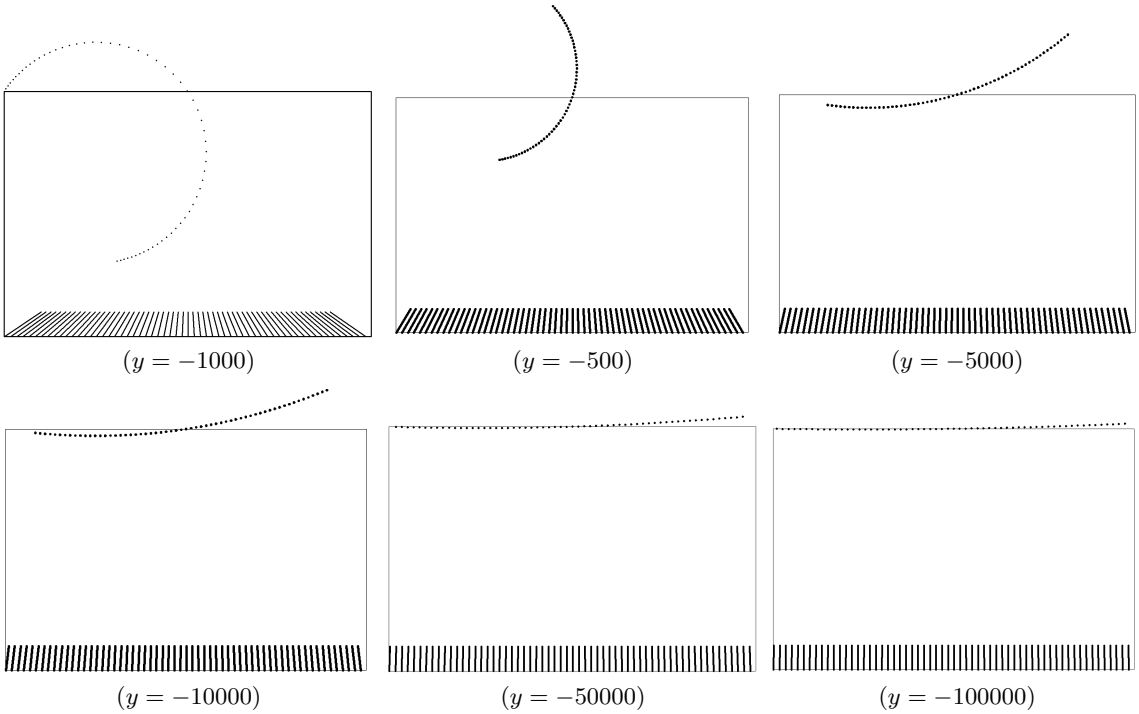



**Fig. 6.11.** Exemple des différentes configurations de la localisation du point de fuite et des segments. La coordonnée en $x$ reste toujours la même, ($x$ = 1500 pixels). Les cercles sont superposés aux segments pour une meilleure compréhension. Dans cette illustration, les segments ne sont pas bruités.

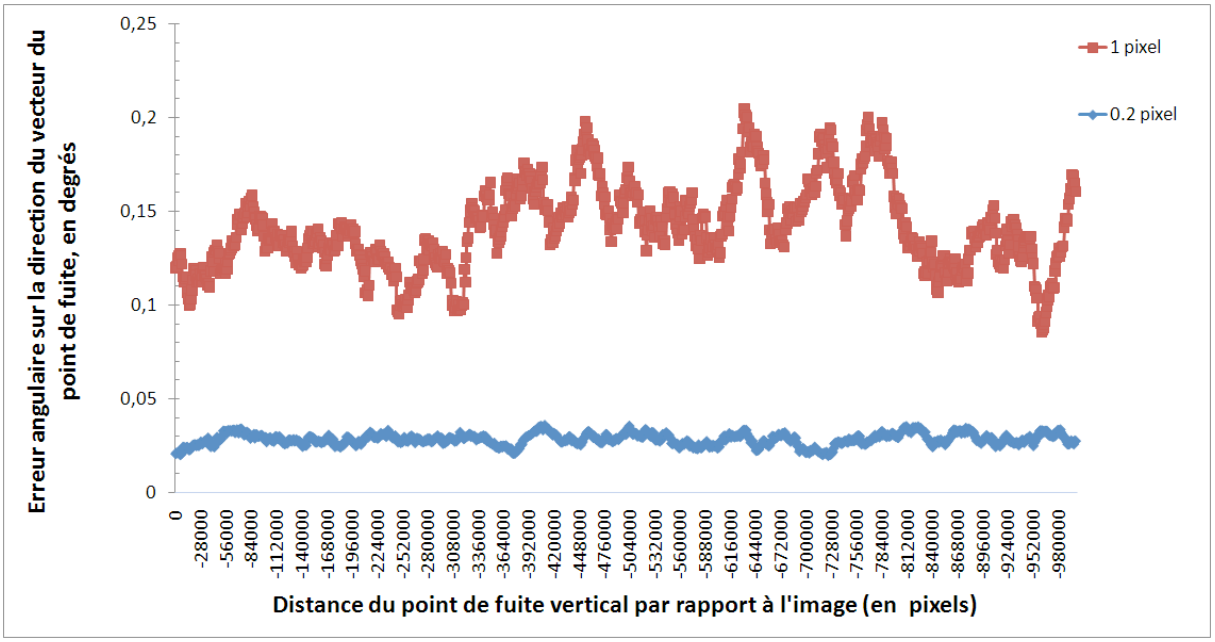



**Fig. 6.12.** Résultats de la comparaison des trois méthodes de détection des points de fuite : méthode des cercles (cf. chapitre 5)

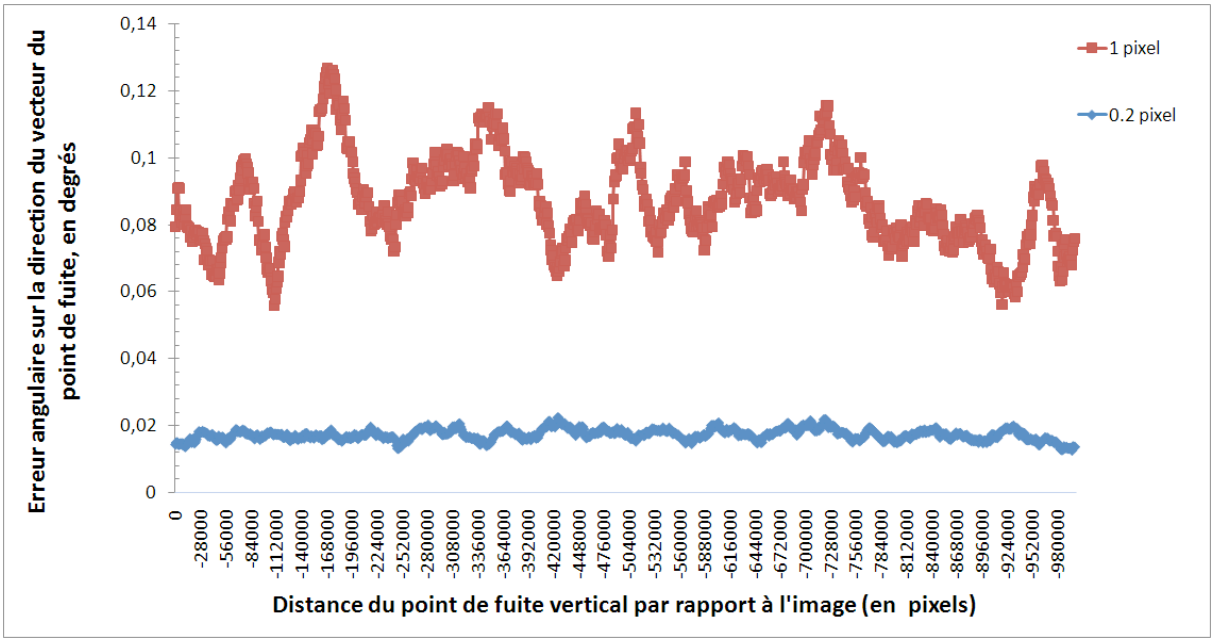



**Fig. 6.13.** Résultats de la comparaison des trois méthodes de détection des points de fuite : méthode des plans (cf. chapitre 6)

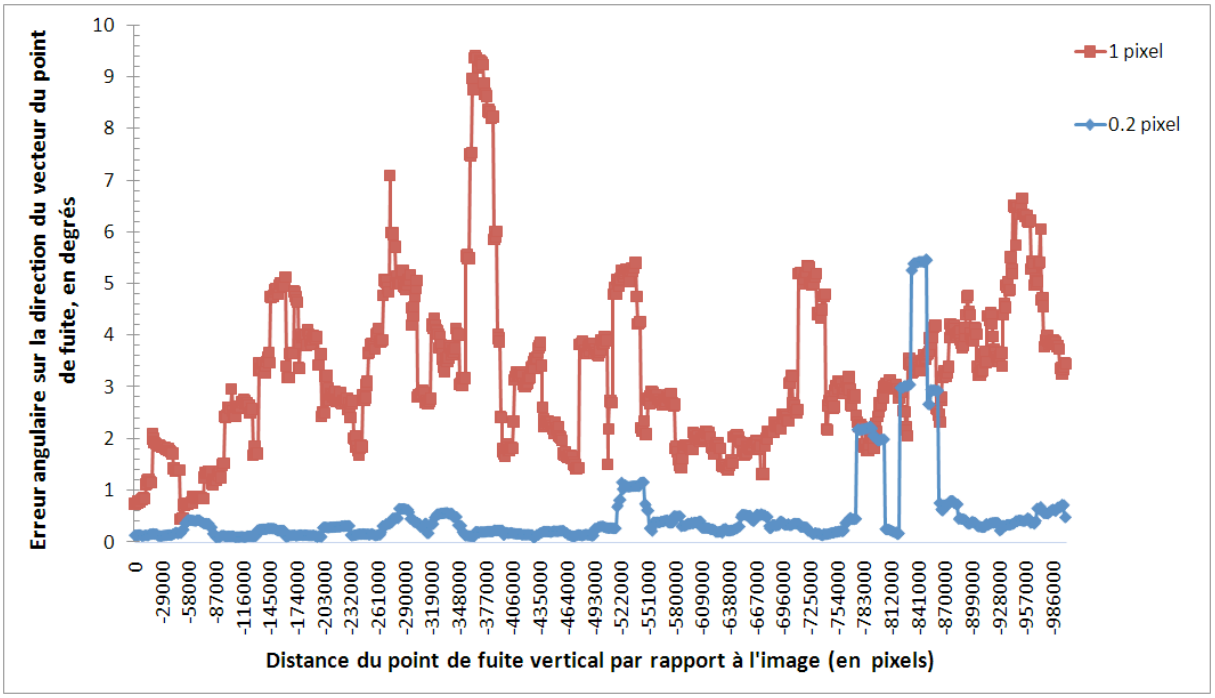



**Fig. 6.14.** Résultats de la comparaison des trois méthodes de détection des points de fuite : méthode d'intersection des segments (cf. [Rother, 2002])

## 6.9 Conclusion

Un nouvel algorithme de détection des points de fuite utilisant la sphère de Gauss, mais sans les inconvénients des processus d'accumulation habituels, a été présenté dans ce chapitre. L'avantage majeur par rapport aux précédents algorithmes est celui de sa rapidité, tout en gardant une excellente précision. La connaissance des paramètres d'étalonnage (y compris la distorsion, sauf si elle est forte au point de perturber l'extraction des segments), n'y est d'ailleurs pas obligatoire. L'amélioration considérable de l'efficacité est due à l'utilisation de la méthode RanSaC, qui introduit une excellente robustesse dans l'estimation sans exiger de connaissance préalables sur les images.

Les chapitres 5 et 6, ont présenté deux méthodes efficaces pour l'extraction des points de fuite dans les images. En plus de la localisation du point de fuite dans celles-ci, les deux algorithmes permettent le calcul de l'incertitude sur ces déterminations. Dans la suite de cette étude, nous allons utiliser les résultats issus de ces algorithmes pour l'orientation des images.

# Partie III

# Estimation de l'orientation et de la localisation d'un ensemble d'images

# 7

# Estimation directe de l'orientation relative à l'aide de 5 points

## Sommaire

## 7.1 Introduction

L'expression algébrique de la condition de coplanarité a été présentée dans l'équation 2.1. Il a été rappelé à cette occasion que les chercheurs de la communauté de vision par ordinateur préféraient une simplification basée sur un regroupement des deux matrices décrivant la translation et la rotation. Dans la présente approche, nous ne procédons pas à cette simplification, et nous proposons donc de garder les équations telles quelles. Bien que l'emploi de la matrice essentielle ait de nombreux avantages, le fait que les paramètres de rotation et translation soient entremêlés engendre des dysfonctionnements, notamment quand la translation est très petite voire nulle. D'un autre coté, il a été démontré que, en théorie, les méthodes de résolutions de l'orientation relative avec l'aide de 5 points ne devraient pas créer d'ambigüité de résolution lorsque tous les points sont coplanaires. Néanmoins, dans [Segvic *et al.*, 2007], une étude approfondie montre clairement les failles des méthodes de 5 points basées sur l'emploi de la matrice essentielle dans les scènes planes.

C'est pour cela que, dans notre étude, nous avons cherché à donner une nouvelle modélisation des équations de coplanarité, où les paramètres de la rotation et de la translation sont des paramètres explicites des équations posées et ne sont pas entremêlés. Dans la partie 7.2, nous modéliserons les équations d'orientation relative afin de construire un système polynomial. Le nouvel algorithme de résolution de l'orientation relative est présenté dans la partie 7.4. Pour conclure, dans la partie 7.5, la présente méthode est comparée à la méthode de résolution, dite des 5 points, de [Stewénius *et al.*, 2006], qui a fait l'objet de publications très récentes, et dont les codes de calcul sont publics [1].

## 7.2 Modélisation algébrique de l'orientation relative

Dans cette partie nous décrivons les différentes possibilités de modélisation de la contrainte de coplanarité, en fonction des différentes modélisations de la rotation et translation.

### 7.2.1 Modélisation de la rotation dans l'espace 3D

Une rotation dans l'espace à 3 dimensions présente 3 degrés de liberté. Il existe plusieurs manières de représenter les rotations. Pendant longtemps, en photogrammétrie analogique, les angles d'Euler ont été employés. Avec la photogrammétrie numérique, il est plus judicieux d'employer une représentation vecteur et angle [Kasser et Egels, 2001]. Dans le cas présent nous cherchons des modélisations qui soient les mieux adaptées à la résolution directe de nos équations polynomiales, c'est-à-dire les moins compliquées possible. Nous présentons plusieurs des modélisations étudiées.

*Représentation à l'aide des quaternions (4 paramètres)*

La manipulation numérique de la matrice rotation est beaucoup plus simple avec les quaternions. La matrice de rotation correspondant au quaternion $Q$ de la forme : $Q =$

1. http://vis.uky.edu/~stewe/FIVEPOINT/

$a + bi + cj + dk$ (où $a$, $b$, $c$, $d$ sont des nombres réels) s'écrit de de la manière suivante :

$$R_Q = \begin{bmatrix} 1 - 2\,c^2 - 2\,d^2 & 2\,bc - 2\,da & 2\,bd + 2\,ca \\ 2\,bc + 2\,da & 1 - 2\,b^2 - 2\,d^2 & 2\,cd - 2\,ba \\ 2\,bd - 2\,ca & 2\,cd + 2\,ba & 1 - 2\,b^2 - 2\,c^2 \end{bmatrix}, \tag{7.1}$$

avec la condition que $|Q| = 1$ c'est-à-dire $\sqrt{a^2 + b^2 + c^2 + d^2} = 1$. Avec cette modélisation, le nombre d'inconnues à estimer pour la matrice de rotation est égal à 4.

*Représentation à l'aide de la formule de Thompson (3 paramètres)*

Moins connue que les autres représentations de la rotation, la rotation de [Thompson, 1959] formalise une matrice de rotation avec 3 inconnues.

$$\frac{1}{\Delta}\begin{bmatrix} \Delta' & -\nu & \mu \\ \nu & \Delta' & -\lambda \\ -\mu & \lambda & \Delta' \end{bmatrix} + \frac{1}{2\Delta}\begin{bmatrix} \lambda \\ \mu \\ \nu \end{bmatrix}\begin{bmatrix} \lambda\,\mu\nu \end{bmatrix} \tag{7.2}$$

avec $\Delta = 1 + \frac{1}{4}(\lambda^2 + \mu^2 + \nu^2)$ et $\Delta' = 1 - \frac{1}{4}(\lambda^2 + \mu^2 + \nu^2)$.

*Représentation à l'aide de la formule de d'Olinde-Rodrigues (3 paramètres)*

Cette représentation modélise la rotation à l'aide d'un vecteur unitaire $\omega$, qui est l'axe de la rotation, et l'angle de rotation $\theta$ autour de celui-ci.

$$R_{ver} = I\cos\theta + \sin\theta\,[\omega]_\times + (1 - \cos\theta)\omega\ {}^t\omega. \tag{7.3}$$

*Représentation à l'aide de la transformée de Cayley (3 paramètres)*

Cette représentation est très employée en robotique [Faugère *et al.*, 2006] et permet de représenter la rotation à l'aide d'une matrice antisymétrique [Cayley, 1846].

Si $A$ est une matrice antisymétrique, $A = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$, et la matrice de rotation est obtenue à l'aide de la transformée de [Cayley, 1846], $(I + A)(I - A)^{-1}$ :

$$\frac{1}{1 + x^2 + y^2 + z^2}\begin{bmatrix} 1 + x^2 - y^2 - z^2 & 2xy - 2z & 2y + 2xz \\ 2xy + 2z & 1 - x^2 + y^2 - z^2 & 2yz - 2x \\ 2xz - 2y & 2x + 2yz & 1 - x^2 - y^2 + z^2 \end{bmatrix}. \tag{7.4}$$

avec comme inconnues : $x$, $y$ et $z$.

### 7.2.2 Modélisation de la translation dans l'espace 3D

La translation $T$, $([T_x, T_y, T_z]^T)$ dans un espace à 3 dimensions a 3 degrés de liberté. Or en orientation relative, l'échelle ne peut être estimée ce qui implique que le vecteur de translation n'a que 2 degrés de liberté. Il est tout à fait possible de contraindre un des éléments du vecteur de la translation, par exemple rajouter comme contrainte que $T_x$ soit égal à 1. Mais ce choix

ne s'avère pas toujours judicieux, surtout en photogrammétrie terrestre où il peut y avoir des angles forts de rotation autour les axes $Y$ et $Z$, ce qui n'est pas le cas en photogrammétrie aérienne. Dans cette optique, la seule contrainte qui semble adaptée à rajouter aux systèmes d'équations est la contrainte de vecteur unitaire pour la translation. Avec cette contrainte, il est démontré qu'il est tout à fait possible de gérer la stéréo avant-arrière (translation en $Z$) ainsi que les faibles translations, cas des images panoramiques par exemple. Contraindre la translation à être unitaire revient à dire que le mouvement se fait à la surface d'une sphère unité. Les différentes façons de modéliser la translation unitaire sont cités ci-dessous.

*Modélisation de la translation avec 3 paramètres*

Une des représentations les plus simples consiste à laisser la translation avec 3 paramètres $T_x, T_y, T_z$ à estimer, et de juste rajouter la contrainte de normalité $T_x + T_y + T_z = 1$.

*Modélisation à l'aide d'une sphère unité*

La base est considérée comme un rayon de la sphère de rayon unité. La translation peut alors être modélisée à l'aide de 2 variables, $u$ et $v$. Nous aurons :

$$\begin{aligned} T_x &= \frac{2u}{u^2+v^2+1} \\ T_y &= \frac{2v}{u^2+v^2+1} \\ T_z &= \frac{u^2+v^2-1}{u^2+v^2+1}. \end{aligned} \tag{7.5}$$

## 7.3 Résolution des différents systèmes polynomiaux exprimant l'orientation relative

Maintenant que nous avons des modélisations explicites pour la rotation et la translation, nous pouvons construire différents systèmes d'équations polynomiales. Dans la construction de tels systèmes, deux éléments sont à prendre en considération : le nombre d'inconnues et le degré du polynôme. Comme il faut prendre en compte simultanément ces deux contraintes, il est nécessaire de faire des compromis. Nous avons donc analysé différents systèmes polynomiaux afin de choisir le mieux adapté. Le choix s'est fait sous forme itérative, c'est-à-dire que pour chaque modélisation choisie, nous avons procédé à la résolution directe à l'aide des outils présentés dans le chapitre 3. Les résolutions polynomiales sont obtenues à l'aide de la librairie Salsa [2]. Dans tous les cas présentés, ce sont les mêmes points homologues qui ont été utilisés.

Nous avons choisi la modélisation la plus efficace en fonction du temps de calcul et du nombre de solutions obtenues. Les temps de calcul mentionnés ci-après sont tous obtenus, bien évidemment, avec la même machine.

Nous présentons ici les résultats obtenus pour les différents systèmes polynomiaux.

2. Salsa, Solvers for ALgebraic Systems and Applications : http://fgbrs.lip6.fr/salsa/

### 7.3.1 Système à 7 équations, 7 inconnues

Dans ce système d'équations, la modélisation de la rotation est faite à l'aide des quaternions, présentée dans le paragraphe 7.2.1. La modélisation choisie pour la translation est celle du paragraphe 7.2.2. Pour un couple de points homologues $a = [xa, ya, f]^T$ et $a' = [xa', ya', f]^T$ ($f$ étant la distance focale), nous obtenons à partir de la condition de coplanarité (equation 2.1, rappel : $\overrightarrow{V_2} \cdot (R\overrightarrow{V_1} \wedge \overrightarrow{T}) = 0$), l'équation suivante :

$$\begin{bmatrix} x_{a'} & y_{a'} & f \end{bmatrix} \begin{bmatrix} 0 & T_z & -T_y \\ -T_z & 0 & T_x \\ T_y & -T_x & 0 \end{bmatrix} \begin{bmatrix} 1-2\,c^2-2\,d^2 & 2\,bc-2\,da & 2\,bd+2\,ca \\ 2\,bc+2\,da & 1-2\,b^2-2\,d^2 & 2\,cd-2\,ba \\ 2\,bd-2\,ca & 2\,cd+2\,ba & 1-2\,b^2-2\,c^2 \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ f \end{bmatrix} = 0. \tag{7.6}$$

En effectuant la multiplication nous avons :

$$\begin{aligned} &(xa'(-T_z(2bc+2da)+T_y(2bd-2ca))+ya'(T_z(1-2c^2-2d^2)-T_x(2bd-2ca))+ \\ &f(-T_y(1-2c^2-2d^2)+T_x(2bc+2da)))xa+(xa'(-T_z(1-2b^2-2d^2)+T_y(2cd+2ba))+ \\ &ya'(T_z(2bc-2da)-T_x(2cd+2ba))+f(-T_y(2bc-2da)+T_x(1-2b^2-2d^2)))ya_2+ \\ &(xa'(-T_z(2cd-2ba)+T_y(1-2b^2-2c^2))+ya'(T_z(2bd+2ca)-T_x(1-2b^2-2c^2))+ \\ &(-T_y(2bd+2ca)+T_x(2cd-2ba)))f=0. \end{aligned} \tag{7.7}$$

5 points homologues donnent 5 équations, auxquelles il faut ajouter les 2 contraintes de normalité, une pour le quaternion ($a^2+b^2+c^2+d^2-1=0$), et l'autre pour la translation ($T_x+T_y+T_z-1=0$). Ce qui en tout donne 7 équations avec 7 inconnues.

Les inconnues de ce système sont : $T_x,\ T_y,\ T_z,\ a,\ b,\ c,\ d$. L'ordre $DRL$ est choisi sur les mônomes.

Remarque : il a été préféré de garder les coordonnées des points, avec explicitement la distance focale. Cependant, on peut très bien se passer de la distance focale, et alors utiliser des coordonnées homogènes.

*Nombre de solutions*

Avec cette modélisation le nombre de solution est égal à 80. Nous rappelons que les éléments pour le calcul du nombre de solutions sont donnés dans le chapitre 3. On peut trouver une symétrie dans celles-ci, due à l'utilisation des quaternions, car $Q(a,b,c,d)$ est égal à $Q'(-a,-b,-c,-d)$.

*Durée de la résolution*

La durée de résolution avec l'aide du module GB et RUR de Salsa est égal à 38.5 secondes.

$$B = \{1, T_z, T_y, T_x, d, c, b, a, T_z^2, T_yT_z, T_xT_z, dT_z, cT_z, bT_z, aT_z, T_y^2, T_xT_y, dT_y, cT_y, bT_y, aT_y, dT_x, cT_x, bT_x, aT_x, d^2, cd, bd, da, c^2, cb, ca, b^2, ba, T_z^3, T_yT_z^2, T_xT_z^2, dT_z^2, cT_z^2, bT_z^2, aT_z^2, T_y^2T_z, T_xT_yT_z, dT_yT_z, cT_yT_z, bT_yT_z, aT_yT_z, dT_xT_z, cT_xT_z, bT_xT_z, aT_xT_z, T_zd^2, T_zcd, T_zbd, T_zda, T_zc^2, T_zcb, T_zca, T_zb^2, T_zba, T_y^3, T_xT_y^2, dT_y^2, cT_y^2, bT_y^2, aT_y^2, T_yd^2, T_z^4, T_yT_z^3, T_xT_z^3, dT_z^3, cT_z^3, bT_z^3, aT_z^3, T_y^2T_z^2, dT_yT_z^2, cT_yT_z^2, bT_yT_z^2, aT_yT_z^2, d^2T_z^2\}. \quad (7.8)$$

### 7.3.2 Système à 6 équations, 6 inconnues

A l'aide de la représentation de [Thompson, 1959] pour la matrice rotation, et en utilisant la modélisation de la translation à 3 paramètres, nous obtiendrons un système d'équations polynomiales avec 6 inconnues et 6 équations. Dans cette modélisation, chaque couple de points homologues donne l'équation suivante :

$$\begin{aligned}(xa(-4T_z\nu - 2T_z\lambda\mu - 4T_y\mu + 2T_y\lambda\nu) + ya(4T_z + T_z\lambda^2 - T_z\mu^2 - T_z\nu^2 + \\ 4T_x\mu - 2T_x\lambda\nu) + f(-4T_y - T_y\lambda^2 + T_y\mu^2 + T_y\nu^2 + 4T_x\nu + 2T_x\lambda\mu))xa' + \\ (xa(-4T_z + T_z\lambda^2 - T_z\mu^2 + T_z\nu^2 + 4T_y\lambda + 2T_y\mu\nu) + ya(-4T_z\nu + 2T_z\lambda\mu - 4T_x\lambda - \\ 2T_x\mu\nu) + f(4T_y\nu - 2T_y\lambda\mu + 4T_x - T_x\lambda^2 + T_x\mu^2 - T_x\nu^2))ya' + (xa(4T_z\lambda - 2T_z\mu\nu \\ + 4T_y - T_y\lambda^2 - T_y\mu^2 + T_y\nu^2) + ya(4T_z\mu + 2T_z\lambda\nu - 4T_x + T_x\lambda^2 + T_x\mu^2 - T_x\nu^2) + \\ f(-4T_y\mu - 2T_y\lambda\nu - 4T_x\lambda + 2T_x\mu\nu))f = 0. \quad (7.9)\end{aligned}$$

Les 5 points homologues donnent 5 équations, auquelles il faut rajouter l'équation de normalité de la translation.

Les inconnues de ce système sont : $T_x,\ T_y,\ T_z,\ \mu,\ \lambda,\ \nu$.

*Nombre de solutions*

A l'aide des notions présentées dans le chapitre 3.4, le nombre de solution est égal à 40

$$B = \{1, T_z, T_y, T_x, \nu, \mu, \lambda, T_z^2, T_yT_z, T_y^2, T_xT_z, T_xT_y, \nu T_z, \nu T_y, \nu T_x, \nu^2, \mu T_z, \mu T_y, \mu T_x, u\nu, \mu^2, \lambda T_z, \lambda T_y, \lambda T_x, \lambda\nu, \lambda\mu, \lambda^2, T_z^3, T_yT_z^2, T_y^2T_z, T_xT_z^2, T_xT_yT_z, \nu T_z^2, \nu T_yT_z, \nu^2T_z, \mu T_z^2, \mu T_yT_z, \mu\nu T_z, \lambda T_z^2, \lambda\nu T_z\}. \quad (7.10)$$

*Durée de la résolution*

La durée de calcul des bases de Gröbner ainsi que le calcul des racines réelles à l'aide de Salsa est égal à 13.8 secondes.

### 7.3.3 Un autre système à 6 équations, 6 inconnues

En utilisant les quaternions et les équations paramétriques de la translation (équation 7.2.2), il est aussi possible d'obtenir un système avec 6 équations et 6 inconnues. Dans ce système, chaque couple de points homologues donne l'équation suivante :

$$\begin{aligned}(xa((-t^2-s^2+1)(2bc+2da)+2s(2bd-2ca))+ya((t^2+s^2-1)(1-2c^2-2d^2)-\\ 2t(2bd-2ca))+f(-2s(1-2c^2-2d^2)+2t(2bc+2da)))xa'+\\ (xa((-t^2-s^2+1)(1-2b^2-2d^2)+2s(2cd+2ba))+ya((t^2+s^2-1)(2bc-2da)-\\ 2t(2cd+2ba))+f(-2s(2bc-2da)+2t(1-2b^2-2d^2)))ya'+\\ (xa((-t^2-s^2+1)(2cd-2ba)+2s(1-2b^2-2c^2))+ya((t^2+s^2-1)(2bd+2ca)-\\ 2t(1-2b^2-2c^2))+f(-2s(2bd+2ca)+2t(2cd-2ba)))f.\end{aligned} \tag{7.11}$$

*Nombre de solutions*

Les bases standard de ce système sont les suivantes :

$$\begin{aligned}B=\{&1,s,t,d,c,b,a,s^2,ts,ds,cs,bs,as,t^2,dt,ct,bt,at,d^2,cd,bd,da,c^2,bc,ca,b^2,ba,s^3,ts^2,ds^2,\\ &cs^2,bs^2,as^2,t^2s,dts,cts,bts,ats,sd^2,scd,sbd,sda,sc^2,sbc,sca,sb^2,sba,t^3,dt^2,ct^2,bt^2,at^2,td^2,\\ &tcd,tbd,tda,tc^2,tbc,tca,tb^2,d^3,cd^2,bd^2,ad^2,c^2d,bcd,acd,b^2d,abd,c^3,bc^2,ac^2,b^2c,abc,b^3,\\ &ab^2,s^4,ts^3,t^2s^2,s^2d^2\}.\end{aligned} \tag{7.12}$$

Il y en a 80, donc ce système a 80 solutions.

*Durée de la résolution*

Le temps de calcul est très élevé, il est égal à 163 secondes.

*Conclusion et choix du système de polynômes*

Dans les paragraphes précédents, ont été vues les différentes expressions de l'orientation relative, à l'aide de la contrainte de coplanarité, et explicitées sous forme de polynômes. En fonction du temps de calcul de résolution de chaque système, ainsi que du nombre de solutions maximales (réelles et complexes), le système le mieux adapté a été choisi. Ce système est celui décrit dans l'équation 7.9. Ce système a 40 solutions, et son temps de calcul est le plus faible, c'est donc ce système qui va être utilisé.

## 7.4 Le nouvel algorithme de résolution de l'orientation relative

Maintenant que tous les éléments de modélisation géométrique et mathématique sont précisés, il est possible de décrire les différentes étapes algorithmiques de la stratégie de détermination de l'orientation relative. Celles-ci sont illustrées dans la Figure 7.1.

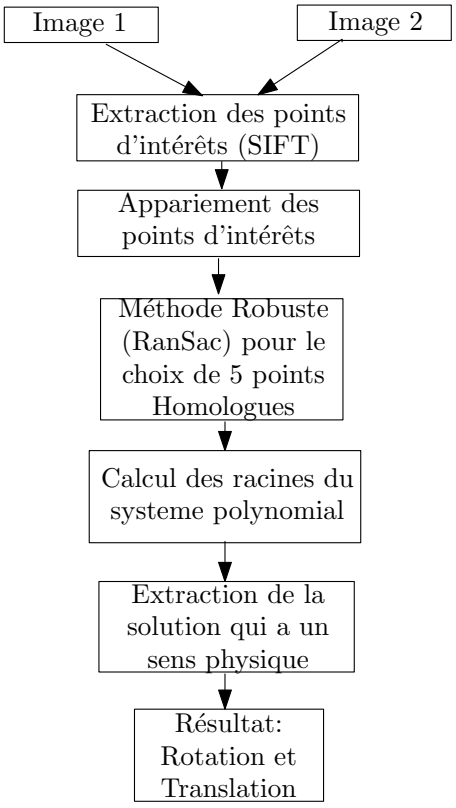



**Fig. 7.1.** Les différentes étapes de l'algorithme de résolution de l'orientation relative.

*Choix robuste de* 5 *points homologues*

Dans le chapitre 2.3, l'extraction automatique des points de liaison avec la méthode SIFT [Lowe, 2004] a été décrite. Il a aussi été vu comment l'appariement, à l'aide de la méthode du kd-tree, permettait l'appariement des points SIFT homolgues. Bien que cette méthode soit très efficace, il reste toujours une petite probabilité que les points appariés soient faux. C'est pour cette raison qu'il est important d'avoir recours à des méthodes d'estimation robuste. Dans la présente application, la méthode statistique, désormais classique, du RanSac est utilisée. L'objectif est de pouvoir estimer les paramètres inconnus, en l'occurrence ici la rotation et la translation, par un nombre minimum de points. Ici donc 5 points sont choisis aléatoirement. A l'aide du modèle (condition de coplanarité) estimé, on évalue quelles autres observations peuvent être considérées comme compatibles avec ce modèle. En fonction de la proportion de points faux (issus de l'appariement initial effectué sur les points SIFT), le nombre d'itérations de ce processus est plus ou moins important. On qualifie finalement la solution qui agrège le plus grand nombre de points possible.

Il a été constaté que le nombre de points appariés à tort était de l'ordre de 5% de la totalité des points. Selon le modèle statistique de la méthode RanSac, il faut donc procéder à 4 tirages pour assurer la probabilité de 0.99 qu'au moins un de ces tirages contienne 5 points correctement appariés. Et plus la proportion de faux points homologues est élevée, plus il faudra de tirages [Hartley et Zisserman, 2004]. D'ailleurs, il est important de noter que cette méthode de filtrage portant sur 5 objets nécessite considérablement moins d'itérations que

si on doit l'appliquer, par exemple, sur 8 objets. Ceci plaide ainsi très directement pour une recherche de méthode de résolution portant sur ce nombre minimal de 5 points.

Dans cette étape 5 points sont donc aléatoirement tirés, avec une distance minimale de 100 pixels entre ces points dans l'image, évitant de prendre en compte des points trop proches. Ces 5 points sont placés dans les équations 7.9. Toutes les racines possibles du polynôme sont calculées à l'aide des outils mathématiques évoqués précédemment (chapitre 3).

*Extraction de la solution qui a le meilleur sens physique*

A l'issue de l'étape précédente, toutes les solutions possibles du système de polynômes sont calculées. Ces solutions sont purement mathématiques. Parmi ces solutions, une et seulement une a un sens correspondant à la géométrie réelle de l'acquisition. Le premier test à effectuer est de vérifier, pour chaque ensemble de solutions (rotation + translation), celles qui permettent d'obtenir, après triangulation, des coordonnées positives. En d'autres termes, on s'assure que les points objets sont bien devant la caméra. Ce premier tri permet de rejeter toutes les configurations où les rayons issus des points homologues se coupent derrière l'image. En général après cette étape la moitié des solutions sont écartées. Ensuite, avec toutes les configurations, on calcule une profondeur moyenne pour tous les points objets. Les fausses solutions donnent presque toujours des axes optiques très convergents, et donc des points qui sont très proches du plan focal. Rappelant que la longueur de la base est prise égale à 1, pour chaque configuration le rapport $\frac{base}{profondeur}$ est calculé. Il est clair que si la profondeur moyenne des points objet est trop petite ce rapport $\frac{B}{H}$ sera très grand, ce qui en terme de configuration photogrammétrique est aberrant. Toutes les configurations qui donneront une profondeur moyenne inférieure à 1 sont rejetées. En général, après cette étape, il ne reste que 2 ou 3 candidats. La dernière étape consiste à garder la solution qui a agrégé le plus grand nombre de points.

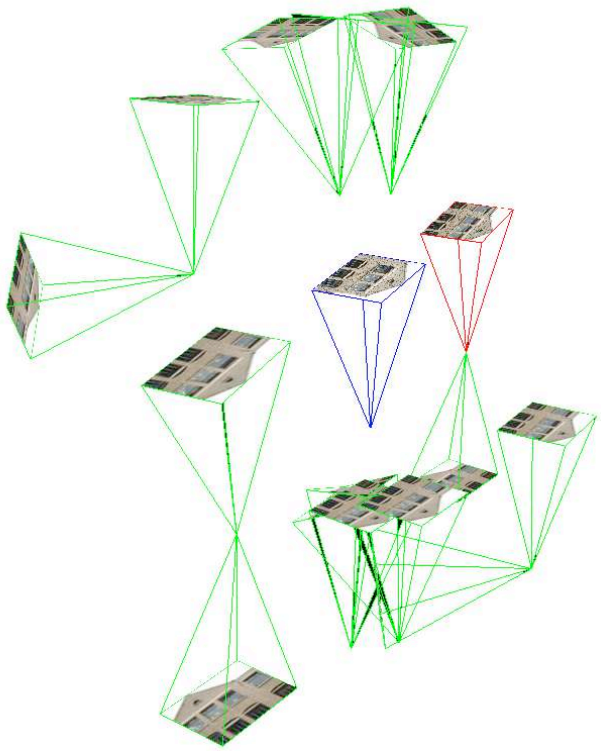


**Fig. 7.2.** Illustration des différentes solutions en vert. En bleu l'image de gauche qui sert de référence, en rouge la meilleure solution extraite parmi les autres solutions.

D'autres méthodes existent pour trouver la bonne solution parmi toutes celles produites par les résolutions directes, mais en général elles consistent à faire intervenir une troisième image [Nistér, 2004]. Dans la méthodologie présentée ici, on n'emploie pas cette solution, et le travail est effectué avec seulement deux images.

## 7.5 Résultats et évaluation

### 7.5.1 Evaluation sur des données de synthèse

#### Configuration de la simulation

Afin de donner des valeurs quantitatives et de tester les performances de la méthode présentée dans ce chapitre, des données synthétiques ont été fabriquées. Les paramètres de simulations sont les mêmes que ceux décrits dans l'article de [Nistér, 2004], à savoir un angle de vue de 45 degrés, une distance à la scène égale à 1. La taille de l'image fait 352 x 288 (CIF).

Différentes configurations ont été traitées :

1. Configuration dite facile : la base entre les deux images a une longueur de 0.3, la profondeur varie entre 0 et 2.

2. Structure plane et base courte (0.1) : dans cette configuration tous les points se situent sur le plan $Z = 2$.
3. Analyse des différentes valeurs de la base dans la détermintion de la rotation et translation.

La procédure de simulation est la suivante :
- 5000 points terrain sont générés de manière aléatoire, selon les trois configuration citées précédemment.
- Les points terrain sont projetés sur les deux plans images à l'aide de l'équation de colinéarité.
- Les coordonnées des points homologues sur chaque image sont bruitées avec un bruit gaussien, d'écart type variant de 0 à 1 pixel, par pas de 0.1 pixel, pour chaque configuration.
- L'angle formé entre le vecteur de la base théorique et la base estimée est une valeur qui est retenue ici pour quantifier l'erreur de l'orientation relative. Et l'écart angulaire entre la vraie rotation et celle estimée nous donne l'erreur d'orientation.
- 100 tirages sont alors effectués pour chaque configuration.

*Remarque : Dans les simulations faites, il est important de préciser que la rotation entre les deux axes optiques est toujours très bien calculée. Ce qui fait la différence entre les différentes méthodes, c'est la précision d'estimation sur l'orientation de la base, et c'est donc le critère d'évaluation qui a été retenu de façon prioritaire [Nistér, 2004].*

*Résultat sur des scènes selon la configuration dite "facile"*

La configuration de la simulation est illustrée dans la Figure 7.3.

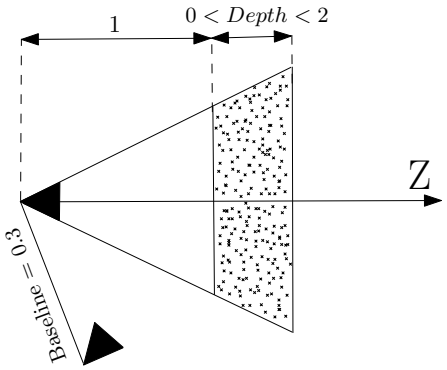



**Fig. 7.3.** Illustration de la simulation : configuration dite "facile".

L'algorithme proposé dans cet article est confronté à la méthode des 5 points de [Stewénius *et al.*, 2006], dans un premier temps, dans une configuration dite facile, c'est-à-dire avec une base de 0.3 et une profondeur qui varie entre 0 et 2. La comparaison est menée en utilisant les codes de calcul téléchargeables sur le site de Stewenius[3].

3. http://vis.uky.edu/~stewe/FIVEPOINT/

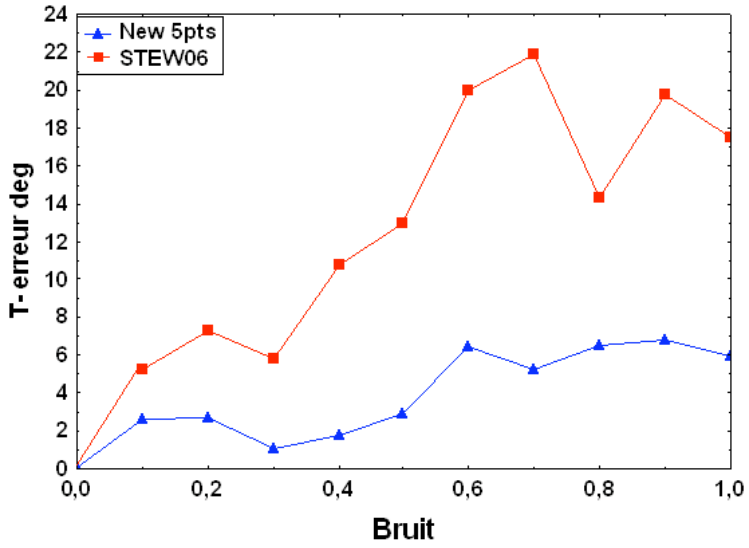



**Fig. 7.4.** Configuration dite "facile", stéréo latérale : erreur angulaire sur l'orientation de la base, en degrés.

Dans la Figure 7.4 le déplacement de la base se fait dans la direction de l'axe des $X$ (donc parallèle à l'objet). On peut voir que le nouvel algorithme permet une bonne détermination de l'orientation de la base (appelée ici T-erreur), l'écart moyen calculé sur une centaine d'essais ne dépassant pas 3 degrés pour un bruit inférieur à 0.5 pixel. Les déterminations apparaissent comme nettement meilleures que celles obtenues en utilisant le code de calcul de Stewenius. Il reste toutefois dans ces courbes un bruit d'échantillonnage assez important, dû à des simulations qui ont dues être limitées à 100 tirages pour chaque situation.

La Figure 7.5 montre des résultats sur une situation de stéréoscopie avant-arrière considérée comme favorable, avec une base de 0.3 suivant l'axe des $Z$. Les résultats du nouvel algorithme sont sensiblement équivalents à ceux évoqués Figure 7.4, on note toutefois que le gain en qualité de détermination par rapport aux calculs de Stewenius est moindre.

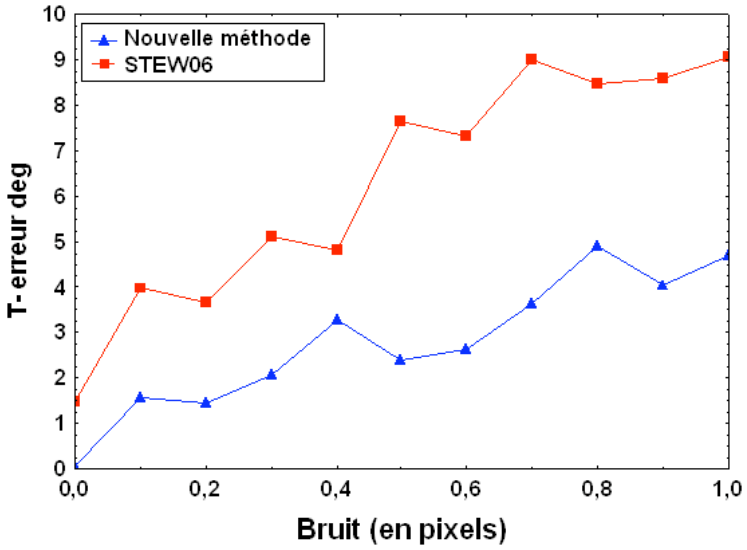



**Fig. 7.5.** Ici la base est selon $Z$ (T-erreur), de valeur 0.3, configuration dite "facile", en stéréoscopie avant-arrière.

*Résultat sur des scènes de structure plane et base courte* (0.1)

Plusieurs surfaces sont connues sous le nom de "dangereuses" [Philip, 1998], la raison de cette appellation est due au fait que si les points choisis pour l'estimation de l'orientation relative se trouvent sur ce genre de surface, la configuration devient dégénérée. Parmi ces surfaces on trouve les surfaces planes, et donc de nombreux cas de photogrammétrie architecturale sont défavorables, car les murs posent évidemment ce type de problème. Dans la suite, une des configurations les plus défavorables a été choisie, illustrée dans la figure 7.6.

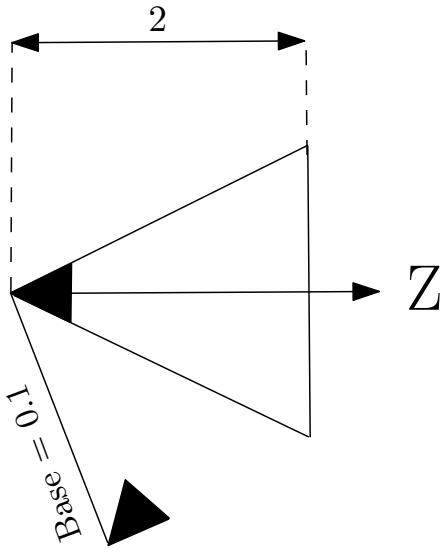



**Fig. 7.6.** Illustration de la simulation : scène plane, base courte.

Le cas examiné est le suivant : les points objet ont une coordonnée $Z$ constante égale à 2. La base est égale à 0.1. Deux cas de géométrie stéréoscopique sont traités. Dans la Figure 7.7,

le déplacement de la base se fait dans direction de l'axe des $X$ (donc parallèle à l'objet). On note que la méthode des 5 points de Nister/Stewenius est très peu robuste à ce genre de configuration. D'ailleurs [Segvic *et al.*, 2007] a montré dans sa publication cette faiblesse de l'algorithme et conclu qu'il vaut mieux dans de tels cas utiliser une homographie. Par contre, avec la méthode présentée ici, ce genre de configuration ne pose aucun problème.

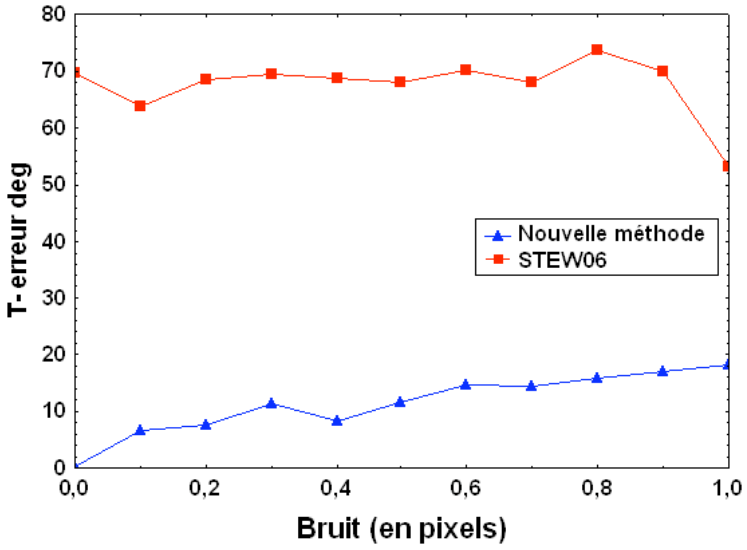



**Fig. 7.7.** Base selon X de valeur 0.1 avec un objet plan. On note que le nouvel algorithme fonctionne de façon satisfaisante, même dans ce cas jugé très défavorable.

Dans la Figure 7.8 la configuration est la même, mais avec une translation selon l'axe des $Z$, toujours de valeur 0.1. Cette fois ci, la nouvelle méthode donne des résultats sensiblement équivalents à celle de Stewenius.

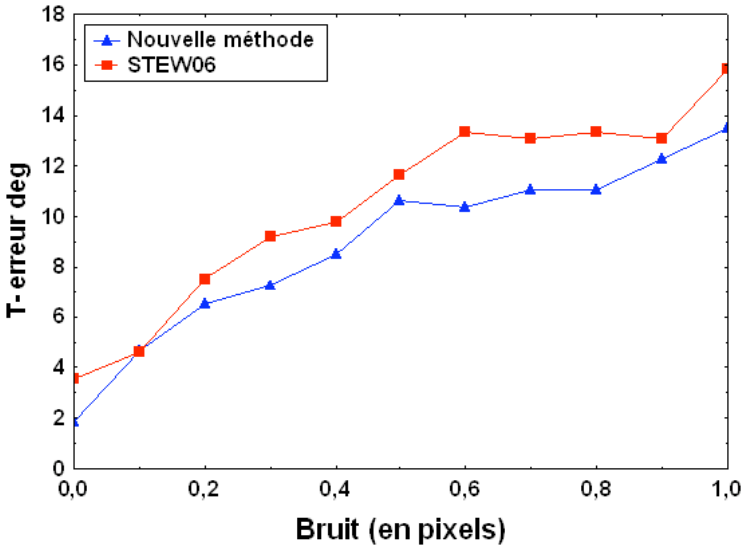



**Fig. 7.8.** Simulations avec une base selon $Z$ de valeur 0.1, avec un objet plan.

*Résultat avec une base nulle, cas d'images panoramiques*

Nous examinons la robustesse de l'algorithme dans un cas où la translation est nulle, cas par exemple des images panoramiques. Il est évident que la translation sera de toute façon estimée de manière incorrecte car dans toutes les équations on l'oblige à avoir une longueur unité. En revanche, la rotation est estimée de manière correcte. Ce bon fonctionnement est dû au fait que, dans les équations, les paramètres inconnus de la rotation et de la translation ne sont pas entremêlés, ce qui n'est pas le cas d'emploi de la matrice essentielle : la configuration de la simulation est illustrée dans la Figure 7.9.

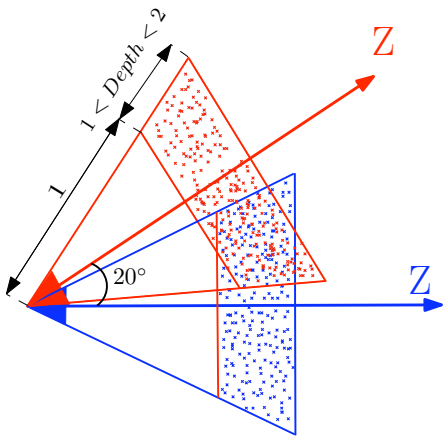



**Fig. 7.9.** Illustration de la simulation : configuration "panoramique".

Les résultats de cette simulation sont présentés dans la figure 7.10.

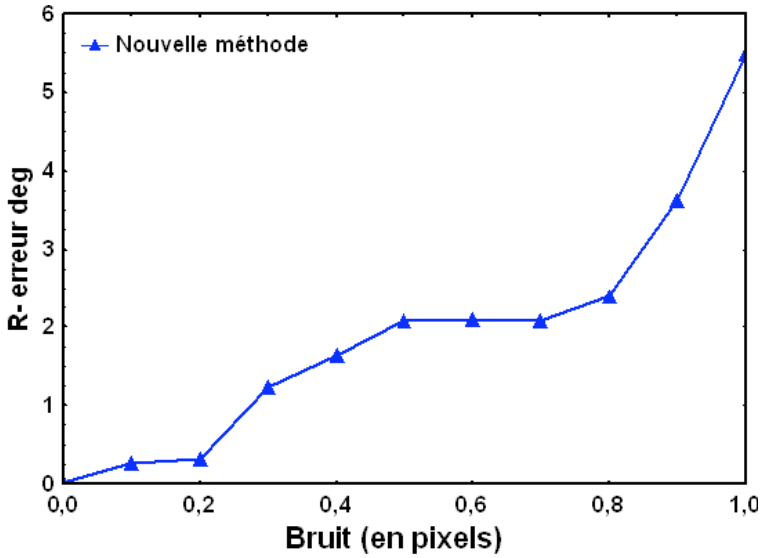



**Fig. 7.10.** Résultats de l'estimation de la rotation en présence de bruit. La longueur de la base est nulle.

*Résultat sur différentes valeurs de la base*

Afin de voir l'impact de la longueur de la base sur la précision de calcul de la translation et de la rotation, celle-ci a fait l'objet de variations allant jusqu'à 0.35, commençant à 0.05.

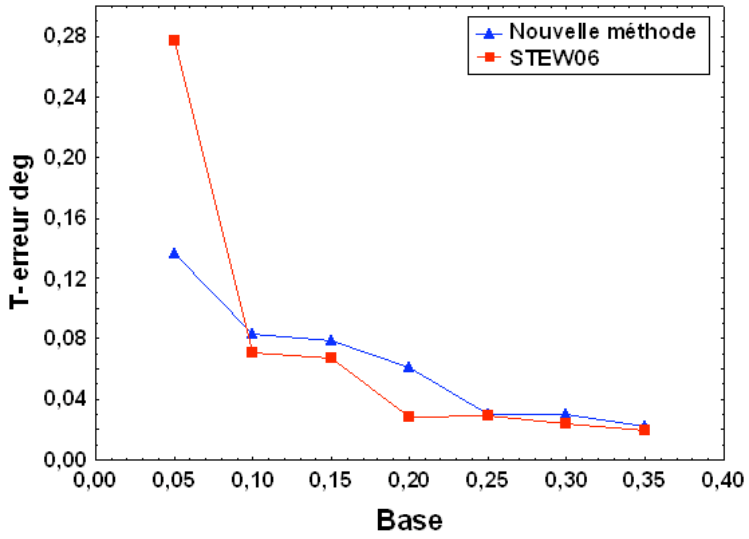



**Fig. 7.11.** Erreurs sur l'orientation de la base, avec la base selon X et un objet non plan, pour des valeurs de base allant jusqu'à 0.35. On note que le nouvel algorithme fonctionne de façon satisfaisante, avec des résultats très proches, voire un peu meilleurs, que ceux de Stewenius.

Puis l'erreur sur la rotation relative entre les deux images est analysée lorsqu'on fait varier la longueur de la base entre 0 et 0.35. On peut constater à partir de la figure 7.11 que même pour une longueur de base quasiment nulle (cas des images panoramiques), la rotation dans les deux méthodes est très bien déterminée.

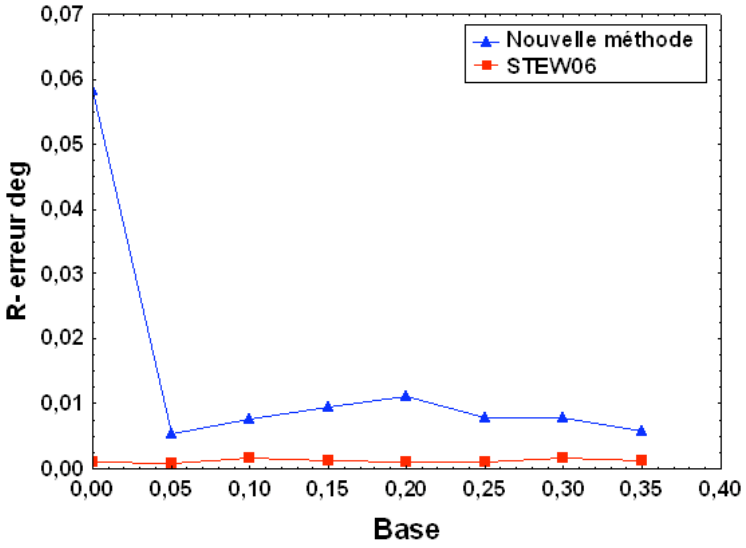



**Fig. 7.12.** La base est selon l'axe des $X$, elle varie jusqu'à 0.35, et c'est l'erreur sur la rotation qui est présentée ici. La méthode de Stewenius est un peu plus précise, mais il faut bien noter que de toutes les façons les résultats sont très bons, restant en dessous de 0.01 degré.

Enfin, on procède à des simulations pour lesquelles la base varie aussi jusqu'à 0.35, mais selon l'axe des $Z$ (stéréoscopie avant-arrière). On note que les résultats, tant pour l'orientation de la base que pour l'orientation, sont tout à fait satisfaisants, l'orientation de la base étant même un peu meilleure que dans la méthode de Stewenius, mais de toutes les façons à un niveau de précision qui ne pose pas de problème (cf Figures 7.13 et 7.14).

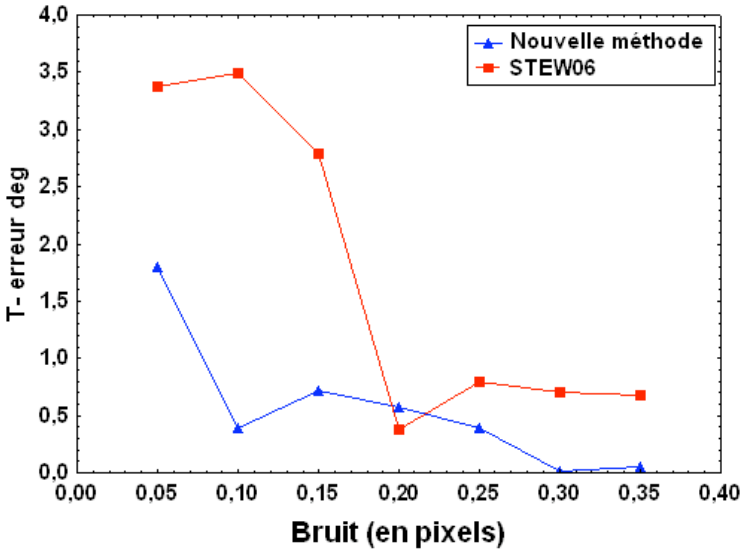



**Fig. 7.13.** Simulations avec une base variable, orientée suivant l'axe des $Z$ : la restitution de l'orientation de la base est satisfaisante (de l'ordre de 1 degré), la nouvelle méthode étant en général bien plus efficace que la méthode de Stewenius.

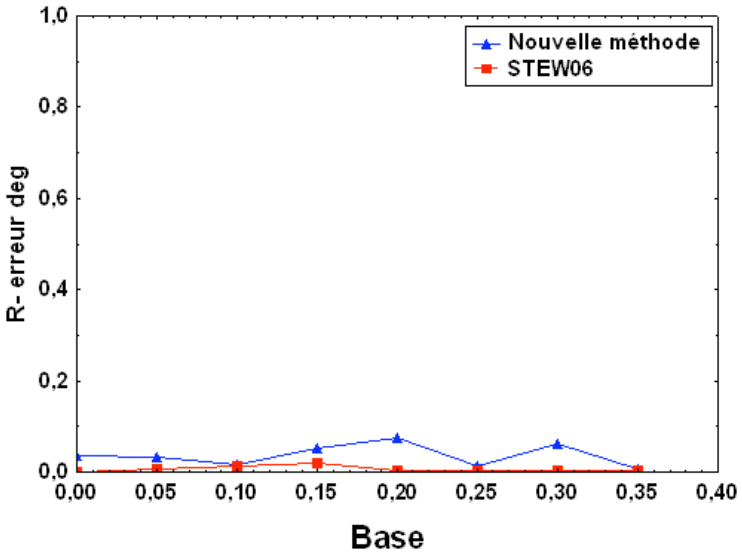



**Fig. 7.14.** Simulations avec une base variable, orientée suivant l'axe des $Z$ : la restitution de la rotation entre les images est très précise (meilleure que 0.01 degré), la nouvelle méthode étant ici un peu moins précise que la méthode de Stewenius, mais à un niveau qui reste d'une précision très satisfaisante.

### 7.5.2 Evaluation sur une base d'images

Afin de fournir un exemple numérique sur de vraies images, nous avons choisi de travailler sur la séquence de 5 images de la base de données en ligne "stones" de l'ISPRS[4]. Dans cette base de données nous disposons de tous les paramètres intrinsèques et externes. L'orientation relative a été calculée dans chaque couple obtenu avec les 5 images, ce qui fait un total de 10 orientations. Les résultats de cette évaluation sont illustrées dans le tableau, où on peut voir la moyenne des erreurs des 10 orientations, sur l'estimation de l'orientation de la base et celle de la rotation.

**Tableau 7.1.** Résultats sur la séquence d'image "stones" de l'ISPRS

| | |
|---|---|
| erreur moyenne sur l'orientation de la base | 5.25° |
| erreur moyenne sur l'orientation de rotation | 1.26° |

**Fig. 7.15.** Quelques images extraites de la base ISPRS "stones"

4. http://www.ipf.tuwien.ac.at/car/isprs/test-data

*Résultat sur des cas réels à titre d'illustration*

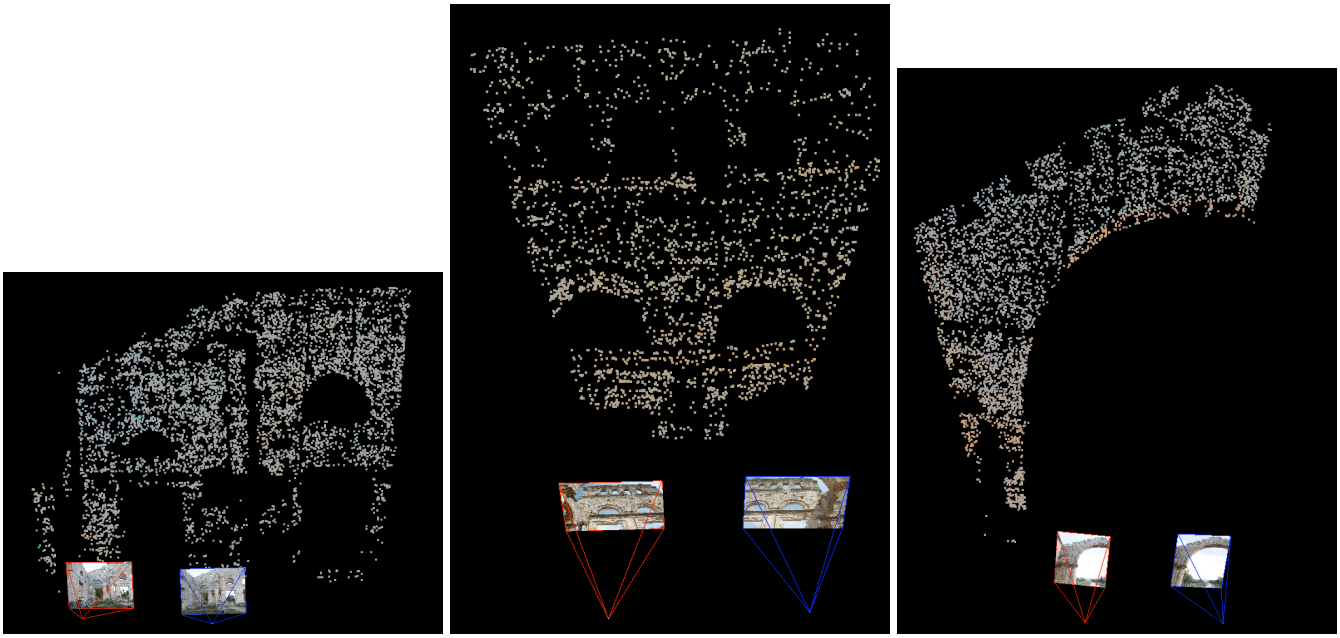


**Fig. 7.16.** Plusieurs résultats sur le Monastère Saint-Siméon en Syrie (crédit photo : Yves Egels)

## 7.6 Conclusions et discussion

Dans ce chapitre, une nouvelle formulation pour le calcul direct des paramètres d'orientation relative (rotation et translation) a été décrite, utilisant le minimum absolu de 5 points, et ne nécessitant aucune valeur approchée. Les résultats obtenus au cours des différentes simulations montrent que l'algorithme est assez peu sensible au bruit. Mais au delà de cette qualité, son principal point fort est la robustesse du calcul dans des cas ou tous les points sont coplanaires, y compris quand la base est parallèle à l'objet imagé. Cette robustesse est bien meilleure que celle des résolutions basées sur la matrice essentielle. Ces travaux ouvrent la voie à des applications en photogrammétrie, qui jusqu'ici ont été utilisées uniquement dans le domaine de la vision par ordinateur. Un des domaines d'emploi prévisibles est celui de la photogrammétrie architecturale, qui est en pleine expansion actuellement compte tenu des besoins de représentations 3D des zones urbaines. En outre, toujours pour des applications photogrammétriques classiques, ce processus de calcul peut être employé pour fournir des valeurs approchées à des calculs d'aérotriangulation, complétant ainsi les outils modernes de prises de vues aériennes tels que les centrales inertielles et les mesures GPS.

# 8

# Orientation relative à partir de 3 points homologues et de la direction verticale

## Sommaire



## 8.1 Introduction

Nous proposons ici une solution efficace au problème de l'orientation relative dans le cadre de systèmes étalonnés. Dans une telle situation, les paramètres intrinsèques de l'appareil-photo, par exemple la distance focale et sa distorsion, sont supposés connus a priori. Dans ce cas l'orientation relative qui lie deux vues est modélisée par 5 inconnues : la matrice de rotation (3 inconnues) et la translation (2 inconnues avec un facteur d'échelle indéterminé). Ces éléments ont été longuement discutés dans le chapitre 7.

Dans ce chapitre, nous utilisons la connaissance de la direction verticale pour résoudre le problème de l'orientation relative pour deux raisons :

1 - l'usage de plus en plus fréquent de dispositifs mixtes micro-mécaniques et inertiels (MEMS-IMU) dans appareils personnels électroniques tels que téléphones intelligents, appareils-photo numériques et centrales inertielles de faible prix. La fusion des capteurs (camera-IMU) n'est pas le but de la présente étude, puisque beaucoup d'auteurs ont déjà montré l'intérêt de les associer [Lobo et Dias, 2003]. Dans les MEMS-IMU la précision en cap (rotation autour de l'axe vertical Z) est moins bonne que le tangage (rotation autour de l'axe X) et le roulis (rotation autour de l'axe Y), ceci étant lié au fait que le champ de la gravité n'a aucun effet sur une rotation autour de l'axe vertical. La nouvelle méthode, présentée ici, tire tout son intérêt d'une combinaison de données inertielles et de l'usage de 3 couples de points homologues qui confortent précisément le point faible de données inertielles.

2 - Aujourd'hui des algorithmes performants basés sur l'analyse d'image sont disponibles, et ils permettent de calculer la direction verticale avec une grande précision. Si nous ne disposons que d'un ensemble d'images calibrées, sans autres mesures externes, nous pouvons déterminer aussi la direction verticale par extraction automatique des points de fuite dans ces images, ce qui a fait l'objet de la partie II.

Un avantage considérable, par rapport à la méthode des 5 points, est celui de la diminution du nombre de points requis, qui descend à la valeur 3. Car, comme on peut le constater dans l'annexe A dans la Figure A.1, plus le nombre de points à tirer est petit, mois on aura de tirages à effectuer, et donc la vitesse de l'algorithme du Ransac sera bien plus élevée.

## 8.2 Systèmes de coordonnées et éléments de la géométrie d'ensemble

C'est le système classique de coordonnées image (cf Figure 8.1) utilisé en vision par ordinateur qui a été choisi [Hartley et Zisserman, 2004]. Dans le système de la camera $(X_{cam},\ Y_{cam},\ Z_{cam})$, le plan focal est $Z_{cam} = F$ , $F$ étant la distance focale. Étant donnée la matrice $K$ de calibration, la vue est normalisée en transformant tous les points par l'inverse de $K$, $\hat{m} = K^{-1}m$, dans lequel $m$ est un point dans le plan image. Donc la nouvelle matrice de calibration devient la matrice identité, $M$ étant un point dans l'espace objet. Dans le reste de cette étude nous supposons que dans toutes les coordonnées des points dans le plan image sont normalisées. Pour un système stéréoscopique, en orientation relative, le centre du système de coordonnées de l'espace objet est le centre optique $C$ de l'image gauche, avec les mêmes directions d'axes. Le système de coordonnées objet est dénoté par $(X_w,\ Y_w,\ Z_w)$. Dans ce système l'axe $Y_w$ est le long de la verticale physique de l'espace objet.

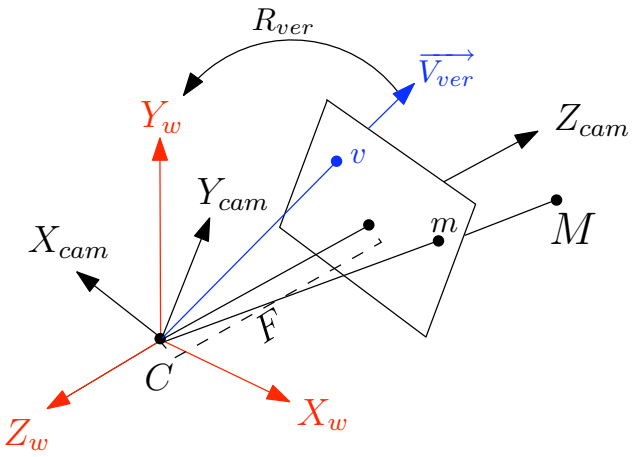



**Fig. 8.1.** Système de coordonnnées employé. Le vecteur $V_{ver}$ est le vecteur du point de fuite vertical et il coupe le plan image en $v$. $R_{ver}$ est définie en 8.3.2

## 8.3 Emploi de la direction verticale pour l'orientation relative

Ce paragraphe concerne l'identification de la verticale selon une approche image. Si cette information est obtenue directement par une mesure physique, l'algorithme décrit dans cette section ne sera pas nécessaire.

Rappelons d'abord que dans la géométrie conique, les lignes parallèles de l'espace de l'objet forment dans l'image un faisceau de droites, concourantes sur les points de fuite. Par conséquent à chaque point de fuite, une direction est associée. La méthode de détection des points de fuite employée dans ce chapitre est celle présentée au chapitre 6.

### 8.3.1 Identification automatique de la direction verticale

Une fois que les différents points de fuite et leurs directions associées sont extraits, nous procédons à une identification automatique du point de fuite qui correspond à la direction verticale, le seul qui nous intéresse ici. Supposons que nous ayons $n$ vecteurs unité des directions de l'espace liant le centre optique aux points de fuite sur le plan image : $\{\overrightarrow{VP_1}, \overrightarrow{VP_2}, \overrightarrow{VP_3}, ..., \overrightarrow{VP_n}\}$. Ces vecteurs sont exprimés dans le système de la caméra. Chaque vecteur est exprimé de la manière suivante : $\overrightarrow{VP} = [V_x\ V_y\ V_z]^t$. Nous choisissons parmi les différents points de fuite extraits, celui qui remplit les conditions suivantes :
1 - le rapport $\frac{V_x}{V_y}$ est minimum,
2 - cette direction est perpendiculaire à celles des autres points de fuite, considérant que la plupart d'entre elles sont horizontales (seul le point de fuite vertical est unique). La prise en compte de la calibration est indispensable ici, et nous écrivons : $90° - \epsilon < \arcsin\left(|\overrightarrow{VP_{vertical}} \otimes \overrightarrow{VP_i}|\right) < 90° + \epsilon$. Par expérience nous avons choisi la valeur de $1°$ pour $\epsilon$. Ces conditions ont été testées sur 250 images de tous types. L'identification automatique du point de fuite vertical a été efficace à 100% . Quelques résultats sont présentés à la Figure 8.2.

La méthode utilisée pour sélectionner le point de fuite vertical est citée ici juste comme une possibilité, beaucoup d'autres peuvent être choisies, par exemple des données MEMS externes peuvent être préférées si disponibles.

*Remarque*

Il est supposé que les photos sont acquises en mode paysage.

### 8.3.2 Orientation absolue autour des axes $Z_w$ et $X_w$

Supposons que $\overrightarrow{V_{ver}}$ est le vecteur qui joint $C$ au point de fuite dans le plan image exprimé dans le système de la caméra, $\overrightarrow{Y_w}(0\ ,1\ ,0)$ étant l'axe $Y$ de l'espace objet (Figure 8.1). Nous cherchons à définir la matrice de rotation qui transforme $\overrightarrow{V_{ver}}$ en $\overrightarrow{Y_w}$, pour cela nous déterminons l'axe de la rotation $\overrightarrow{\omega}$ et l'angle $\theta$ de la rotation autour de cet axe de la façon suivante : $\overrightarrow{\omega} = \overrightarrow{V_{ver}} \otimes \overrightarrow{Y_w}$, après simplification et normalisation $\overrightarrow{\omega} = [\frac{V_z}{d}\ ,0\ ,\frac{-V_x}{d}]$, ou $d = \sqrt{V_z^2 + V_x^2}$, après simplification $\theta = \arccos(V_y)$. Utilisant la formule d'Olinde-Rodrigues nous obtenons la matrice de rotation suivante :

$$R_{ver} = I\cos\theta + \sin\theta\,[\omega]_\times + (1-\cos\theta)\omega\omega^T. \tag{8.1}$$

Cette rotation ($R_{ver}$) est appliquée alors à toutes les coordonnées $2D$ des points obtenus dans chaque image, donc : $\hat{m}$ est remplacé par $R_{ver}\hat{m}$.

## 8.4 Simplification et reformulation de la contrainte de coplanarité

Rappelons en premier que pour une paire de points homologues $\hat{m^1}$ et $\hat{m^2}$ d'une caméra parfaite, la contrainte sur ces 2 points est exprimée par l'équation de coplanarité :

$$\begin{bmatrix}\hat{m_x^2}\ \hat{m_y^2}\ 1\end{bmatrix} E \begin{bmatrix}\hat{m_x^1}\\ \hat{m_y^2}\\ 1\end{bmatrix} = 0. \tag{8.2}$$

où $E$ est la matrice essentielle 3x3 de rang 2 [Hartley et Zisserman, 2004]. Nous pouvons aussi exprimer cette contrainte par l'équation 8.3.

$$\begin{bmatrix}\hat{m_x^2}\ \hat{m_y^2}\ 1\end{bmatrix} \begin{bmatrix}0 & T_z & -T_y\\ -T_z & 0 & T_x\\ T_y & -T_x & 0\end{bmatrix} R \begin{bmatrix}\hat{m_x^1}\\ \hat{m_y^2}\\ 1\end{bmatrix} = 0. \tag{8.3}$$

Cependant, si nous appliquons la rotation ($R_{ver}$) obtenue dans l'équation 8.1 à tous les points homologues, avant que nous ne tenions compte de cette contrainte (équation 8.3), la rotation $R$ est exprimée de façon plus simple, puisqu'il reste seulement un paramètre à estimer, l'angle $\phi$ autour de l'axe $Y$ (axe vertical). Donc :

**Fig. 8.2.** Quelques résultats sur l'extraction et l'identification automatique des lignes et directions verticales. Les lignes détectées sont illustrées en rouge

$$R_\phi = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \tag{8.4}$$

Nous remplaçons $\cos\phi$ par $(1-t^2)/(1+t^2)$ et $\sin\phi$ par $2t/(1+t^2)$. La nouvelle équation de coplanarité sera simplifiée ainsi :

$$\begin{aligned} (&-2\hat{m_x^2}T_y t + \hat{m_y^2}(T_z(1-t^2) + 2T_x t) - \\ &\hat{m_z^1}T_y(1-t^2))\hat{m_x^1} + (\hat{m_x^2}(1+t^2)T_z + \\ &\hat{m_z^2}(1+t^2)T_x)\hat{m_y^1} + (\hat{m_x^2}T_y(1t^2) + \\ &\hat{m_y^2}(2T_z t - T_x(1-t^2)) - 2\hat{m_z^2}T_y t)\hat{m_z^1} = 0. \end{aligned} \tag{8.5}$$

Nous définissons pour 3 paires de points homologues les équations 8.5, et nous les nommons $\{f_2, f_3, f_4\}$. Les inconnues restantes sont par conséquent $T_x, T_y, T_z$ et $t$. La base a seulement 2 degrés de liberté, parce que le modèle n'a pas d'échelle. Par conséquent il est nécessaire, ou bien de fixer une composante de la base à 1, ou d'ajouter la contrainte de normalité. Nous choisissons cette dernière : $f_1 \equiv T_x^2 + T_y^2 + T_z^2 - 1 = 0$. L'avantage est que ceci permet d'obtenir un modèle plus général. Nous avons par conséquent un système de 4 équations polynomiales de degré 3 $\{f_1, f_2, f_3, f_4\}$. Maintenant nous allons décrire la résolution directe de ce système polynomial en utilisant les bases de Gröbner.

## 8.5 Résolution à l'aide des bases de Gröbner

Toutes les définitions necéssaires pour comprendre l'utilisation des bases de Gröbner ont été détaillées dans le chapitre 3.

### 8.5.1 Construction du solveur algébrique spécifique

Dans cette section nous décrivons un algorithme général pour calculer la base de Gröbner du système de polynômes défini dans la section 8.3. Il faut bien noter que quand les coordonnées des points en entrée changent, seuls les coefficients des polynômes changent. Donc, en utilisant l'approche de [Lazard, 1983] (cf. 8.5), nous construisons une matrice de Macaulay (et nous pouvons la calculer directement quand les coordonnées des points changent), et une élimination de Gauss sur cette matrice donne la base de Gröbner de l'idéal.

Soit $f_1, \ldots, f_4 \in \mathbb{C}[T_x, T_y, T_z, t]$ le système de polynômes comme défini en 8.4. Soit $I = \langle f_1, \ldots, f_4 \rangle$.

Notre premier travail est de choisir un "bon" ordre. Comme bon ordre monomial, nous voulons dire un ordre monomial pour lequel le degré maximum atteint dans le calcul de la base de Gröbner est minimum. Et quant à la complexité, nous cherchons un ordre monomial pour lequel le calcul présente la moindre complexité. Nous choisissons l'ordre DRL parce qu'il fournit typiquement les calculs de base de Gröbner les plus rapides.

Ici, nous étudions deux variantes de cet ordre monomial. En premier, nous considérons DRL$(T_x, T_y, T_z, t)$. Nous calculons le degré maximum des monômes qui apparaissent dans le calcul de la base de Gröbner de $I$ pour cet ordre monomial. Pour ceci, nous homogénéisons les $f_i$ par rapport à une variable auxiliaire $h$ et nous calculons la base de Gröbner du système homogénéisé pour DRL$(T_x, T_y, T_z, t, h)$. Le degré maximal des éléments de cette base est 6 et par conséquent le degré maximal des monômes qui paraissent dans le calcul de la base de Gröbner de $I$ sera 6.

Considérons un autre ordre monomial. Pour ceci, nous rappelons en premier la définition de l'ordre monomial de degré par bloc. Soit $X$ et $Y$ deux ensembles de variables et $<_X$ (resp. $<_Y$) un rangement de monômes défini sur $X$ (resp. $Y$). L'ordre monomial de degré par bloc de $<_X$ et $<_Y$ est un rangement de monômes défini sur les monômes en $X$ et $Y$ et nous le dénotons par $<$. Soient $m = xy$ et $m' = x'y'$ deux monômes, où $x$ et $x'$ (resp. $y$ et $y$') sont des monômes en $X$ (resp. $Y$). Nous définissons que $m < m'$ si $deg(m) < deg(m')$, ou $deg(m) = deg(m')$ et $x <_X x'$ ou $deg(m) = deg(m')$ et $x = x'$ et $y <_Y y'$. Maintenant, considérons l'ordre de degré par bloc de DRL$(Tx, Ty, Tz)$ et DRL$(t)$. En utilisant la technique précitée nous pouvons calculer le degré maximal des monômes qui paraissent dans le calcul de la base de Gröbner par rapport à cet ordre monômial, qui est 8.

Donc le premier ordre monomial est meilleur. Notons que pour quelques systèmes un ordre de degré par bloc de DRL est mieux qu'un DRL simple. Nous avons testé quelques autres ordres monomiaux, et il apparaît qu'un DRL simple est le meilleur.

Notre deuxième problème est de construire $M_6(f_1, \ldots, f_4)$, soit donc $M$. Pour calculer une telle matrice, nous devons trouver les produits $mf_i$, de sorte qu'une élimination de Gauss sur la matrice de la représentation de ces produits nous mène à la base de Gröbner de $I$. Pour ceci, nous utilisons le degré maximum atteint dans le calcul de la base de Gröbner, qui est 6. Nous considérons tous les produits $mf_i$ où le $m$ est un monôme de degré au plus $6 - \deg(f_i)$. Cela donne 175 polynômes. Parmi eux, il y a des produits qui sont utiles pour construire $M$. En utilisant le programme en MAPLE suivant, nous pouvons choisir ceux qui sont utiles :

```
L:=NULL:
AA:=A:
for i from 1 to nops(A) do
      unassign('p');
      X:=AA:
      member(A[i], AA, 'p');
      AA:=subsop(p=NULL,AA):
      if IsGrobner(Macaulay(AA)) then
          L:=L,i;
      else
          AA:=X:
      fi:
od:
```

où `IsGrobner` est un programme pour tester si un ensemble de polynômes forme une base de Gröbner pour I ou non, et `Macaulay` est un programme qui exécute une élimination de Gauss sur la représentation matricielle d'un ensemble de polynômes. Cela donne 65 polynômes de degré au plus 6. Dans ce cas-là, $M$ a une dimension de $65 \times 77$. Voici la liste des 65 polynômes qui ont été trouvés ainsi.

$$
\begin{gathered}
f_4, tf_4, T_zf_4, T_yf_4, T_xf_4, tT_zf_4, tT_yf_4, tT_xf_4, \\
T_zT_yf_4, T_zT_xf_4, T_y^2f_4, T_yT_xf_4, T_x^2f_4, tT_zT_yf_4, \\
tT_zT_xf_4, tT_y^2f_4, tT_yT_xf_4, tT_x^2f_4, f_3, tf_3, T_zf_3, \\
T_yf_3, T_xf_3, tT_zf_3, tT_yf_3, tT_xf_3, T_zT_yf_3, T_zT_xf_3, \\
T_y^2f_3, T_yT_xf_3, T_x^2f_3, tT_zT_xf_3, tT_y^2f_3, tT_yT_xf_3, \\
tT_x^2f_3, f_2, tf_2, T_zf_2, T_yf_2, T_xf_2, tT_zf_2, tT_yf_2, \\
tT_xf_2, T_zT_yf_2, T_zT_xf_2, T_y^2f_2, T_yT_xf_2, T_x^2f_2, \\
tT_zT_xf_2, tT_y^2f_2, tT_yT_xf_2, tT_x^2f_2, f_1, tf_1, T_zf_1, \\
T_yf_1, T_xf_1, t^2f_1, tT_yf_1, tT_xf_1, t^3f_1, t^2T_yf_1, \\
t^2T_xf_1, t^3T_yf_1, t^3T_xf_1
\end{gathered}
$$

Grâce à la propriété selon laquelle la division par l'idéal $I$ est bien définie quand nous le faisons par rapport à une base de Gröbner de $I$, nous pouvons considérer l'espace de tous les restes sur division par I (cf. Cox *et al.* [2007a]). Cet espace est appelé *l'anneau quotient* de $I$, et nous le dénotons par $A = \mathbb{C}[Tx, Ty, Tz, t]/I$. C'est une propriété classique, que si $I$ est racine, alors le système $f_1 = \cdots = f_4 = 0$ a un nombre fini de solutions $N$ si et seulement si la dimension de $A$ en tant qu'espace $\mathbb{C}$-vectoriel est $N$ (cf. Cox *et al.* [2007a], proposition 8 page 235). On peut vérifier facilement par la fonction `IsRadical` de MAPLE que $I$ est racine. Une base pour $A$ en tant qu'espace vectoriel peut être obtenue en utilisant $\text{in}(I)$ par (Cox *et al.* [2007a], théorème 6, page 234)

$$B = \{m \mid m \text{ est un monôme et } m \notin \text{in}(I)\}$$

Par le calcul d'une base de Gröbner de $I$, nous pouvons calculer $\text{in}(I)$ qui est égal à $\text{in}(I) = \langle T_x, T_y, T_z^2, t^6\rangle$ et donc l'ensemble

$$B = \{1, t, t^2, t^3, t^4, t^5, T_z, T_zt, T_zt^2, T_zt^3, T_zt^4, T_zt^5\}$$

est une base pour $A$ comme espace vectoriel sur $\mathbb{C}$.

Par conséquent, nous pouvons conclure que le système $f_1 = \cdots = f_4 = 0$ a 12 solutions. Notons que nous avons obtenu ces résultats pour un jeu particulier de coordonnées de points en entrée. Discutons maintenant ces résultats pour tout ensemble de points. Considérons $\hat{m}_x^1, \ldots, \hat{m}_z^3$ comme étant les paramètres, et $I$ comme un idéal de l'anneau $\mathbb{C}(\hat{m}_x^1, \ldots, \hat{m}_z^3)[T_x, T_y, T_z, t]$.

Soit $G$ une base de Gröbner de

$$I \subset K[\hat{m_x^1}, \ldots, \hat{m_z^3}][T_x, T_y, T_z, t]$$

pour un ordre monomial de degré par bloc de DRL$(Tx, Ty, Tz, t)$ et DRL$(\hat{m_x^1}, \ldots, \hat{m_z^3})$, et soit $C_i$ le produit des coefficients initiaux de ses éléments considérés comme polynômes en $Tx, Ty, Tz, t$, avec ses coefficients dans $K[\hat{m_x^1}, \ldots, \hat{m_z^3}]$.

C'est un résultat classique, si $C_i(\hat{m_x^1}, \ldots, \hat{m_z^3}) \neq 0$, alors la substitution des paramètres par leur valeurs dans $G_i$ donne une base de Gröbner de $I$ (cf. [Cox *et al.*, 2007a] p. 288 par exemple). Donc toutes propriétés géométriques de l'idéal obtenues pour des points particuliers (que nous avons étudiés ci-dessus) sont vraies pour presque tout idéal obtenu par un autre jeu de points, et nous disons que ces propriétés sont *génériquement* vraies.

Par conséquent, tout idéal ainsi obtenu est racine, son idéal initial est égal à $\langle T_x,\ T_y,\ T_z^2,\ t^6 \rangle$, et son système correspond a 12 solutions. Nous rappelons ici brièvement la méthode de la valeur propre que nous utilisons pour résoudre le système $f_1 = \cdots = f_4 = 0$, cf. [Cox *et al.*, 1998], page 56 pour plus de détails. Pour tout $f \in \mathbb{C}[T_x, T_y, Tz, t]$ nous notons par $[f]$ le coset de $f$ dans $A$. Nous définissons $m_f : A \longrightarrow A$ par la règle suivante :

$$m_f([g]) = [f].[g] = [fg] \in A.$$

Donc, l'idéal engendré par les $f_i$ est de dimension zéro, donc $A$ est un espace vectoriel de $\mathbb{C}$ de dimension finie, et nous pouvons alors représenter $m_f$ par une matrice qui est appelée la *matrice action* de $f$. Pour tout $i$, si nous prenons $f = x_i$, alors les valeurs propres des $m_{x_i}$ sont les $x_i$ coordonnées des solutions du système. En utilisant ces valeurs propres pour chaque $i$, et un test pour vérifier si un $n - uplet$ de ces valeurs propres annule ou pas les $f_i$, nous pouvons trouver les solutions du système. Une méthode plus efficace est d'utiliser les vecteurs propres. Soit $f$ une forme linéaire générique dans $A$, alors nous pouvons prendre directement toutes les solutions du système parmi les vecteurs propres de $m_f$, cf. [Cox *et al.*, 1998], page 64.

## 8.6 Calcul de l'orientation relative finale

Après la résolution du système polynomial, et l'obtention des paramètres $T_x$ , $T_y$ , $T_z$ et $t$, il est possible de calculer l'orientation relative finale entre les deux images. Si nous supposons que $R^1_{ver}$ est la matrice rotation définie dans la section 8.3.2 pour l'image 1, $R^2_{ver}$, idem pour l'image 2, et $R_\phi$ la matrice de la rotation définie par $t$ (équation 8.4), l'orientation relative définitive entre les images 1 et 2 est :

$$\begin{aligned} R_{final} &= {R^2_{ver}}^t\ R_\phi R^1_{ver}, \\ \overrightarrow{T_{final}} &= {R^2_{ver}}^t\ \overrightarrow{T},\ avec\ \overrightarrow{T} = [Tx, Ty, Tz]^t. \end{aligned} \tag{8.6}$$

## 8.7 Tests expérimentaux

La précision du calcul de l'orientation relative, en utilisant un point de fuite vertical et 3 points de liaison, vient de trois facteurs :
1 - la précision de la résolution polynomiale des paramètres de la translation ($T_x,\ T_y,\ T_z$), et de la rotation autour de l'axe $Y$ en utilisant les bases de Gröbner,
2 - la précision géométrique de l'estimation de la direction verticale,
3 - la précision de l'algorithme sur les points de liaison en présence de bruit.
Pour évaluer ces différents aspects, nous avons dans un premier temps travaillé sur des données synthétiques, en 8.7.1, puis nous avons utilisé de vraies données, en 8.7.2.

### 8.7.1 Performances en présence de bruit

Dans cette section, la performance de la méthode des 3 points en présence de bruit a été étudiée et comparée à l'algorithme de 5 points [Stewénius *et al.*, 2006] en utilisant le logiciel fourni par les auteurs [Stewenius, 2005]. Le dispositif expérimental employé est semblable à [Nistér, 2004]. La distance au volume de la scène est utilisée comme unité de mesure, et la longueur de la ligne de base est de 0.3. L'écart-type du bruit est exprimé en pixels, pour une image 352 x 288, avec $\sigma = 1.0$ pixel. Le champ de vue est égal à 45 degrés. La profondeur varie entre 0 et 2. Deux valeurs différentes de translation ont été traitées, une en X (mouvement transverse) et une en Z (mouvement avant-arrière). Les tests impliquent 2500 tirages aléatoires de points homologues. Pour chaque tirage, nous determinons l'angle entre la ligne de base estimée et le vecteur de la ligne de base vraie. Cet angle est appelé ici erreur translationnelle, et est exprimée en degrés. Pour l'estimation de l'erreur sur la matrice de la rotation, l'angle de $(R_{true}^T R_{estimate})$ est calculé, et la valeur moyenne résultant des 2 500 tirages aléatoires pour chaque niveau de bruit est donnée. D'après les figures 8.3, 8.4, 8.5 et 8.6, nous voyons que l'algorithme de 3 points est plus robuste, si on tient compte des erreurs causées par le bruit, tant en mouvement transverse qu'en mouvement avant-arrière, pour l'estimation de la rotation et de la translation.

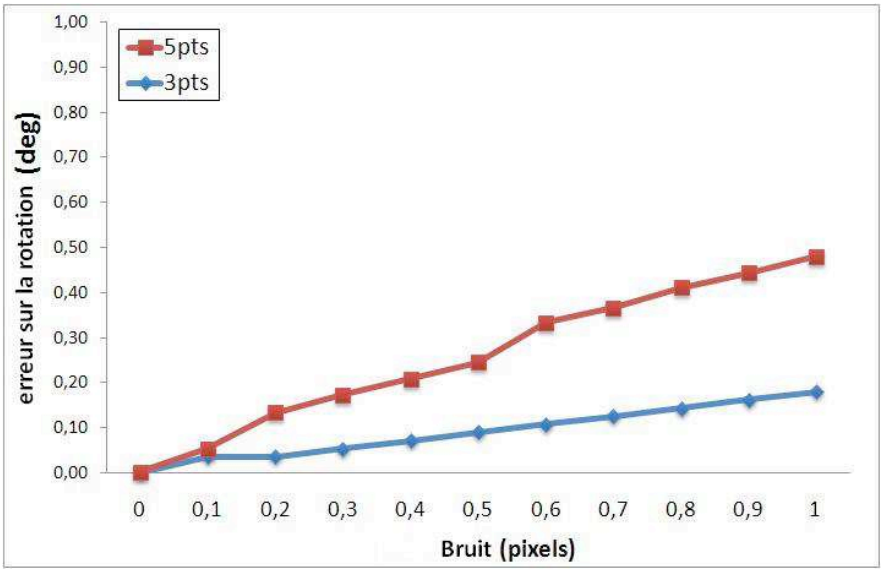



**Fig. 8.3.** Erreur sur la rotation (en degrés, mouvement transverse).

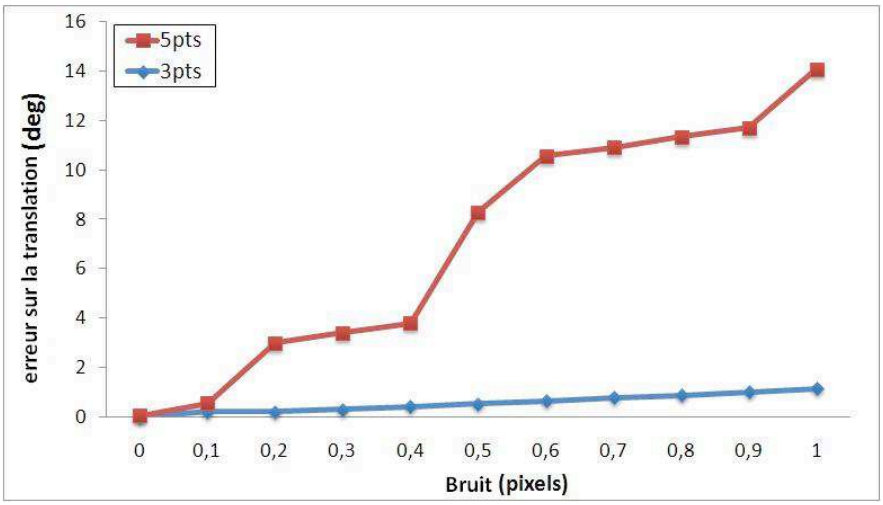



**Fig. 8.4.** Erreur sur l'orientation de la base (en degrés, mouvement transverse).

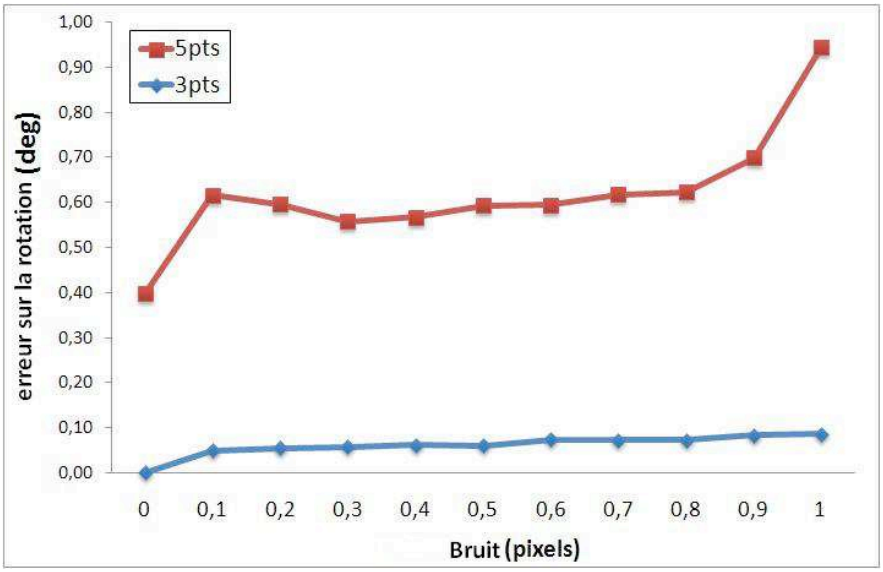



**Fig. 8.5.** Erreur sur la rotation (en degrés, mouvement avant-arrière)

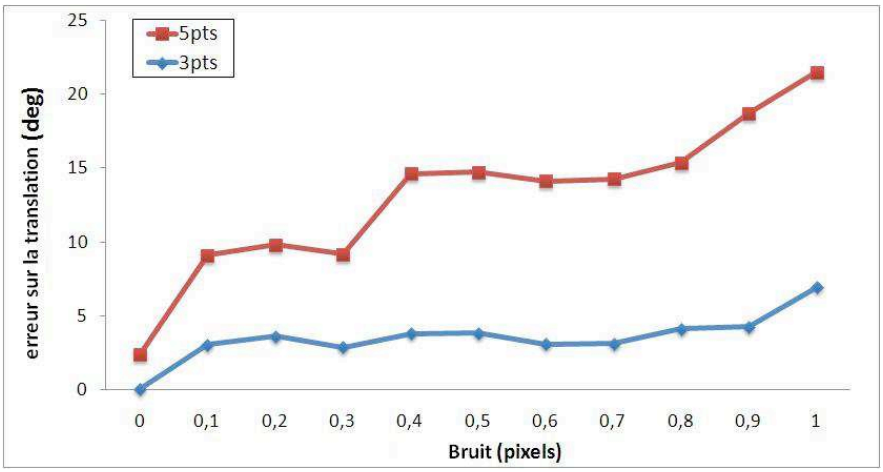



**Fig. 8.6.** Erreur sur l'orientation de la base (en degrés, mouvement avant-arrière).

Maintenant, nous comparons l'algorithme des 3 points et l'algorithme des 5 points sur une scène plane. Dans cette configuration tous les points de la scène l'espace objet ont le même $Z$ (ici égal à 2). Les résultats pour l'estimation de la rotation (Figure 8.7) montrent que les deux algorithmes fournissent une bonne détermination de la rotation, mais la méthode des 3 points donne des résultats bien meilleurs que celle des 5 points pour la détermination de l'orientation de la base en mouvement transverse (Figure 8.8). Cette faiblesse de l'algorithme des 5 points pour une scène plane a été discutée dans [Segvic *et al.*, 2007].

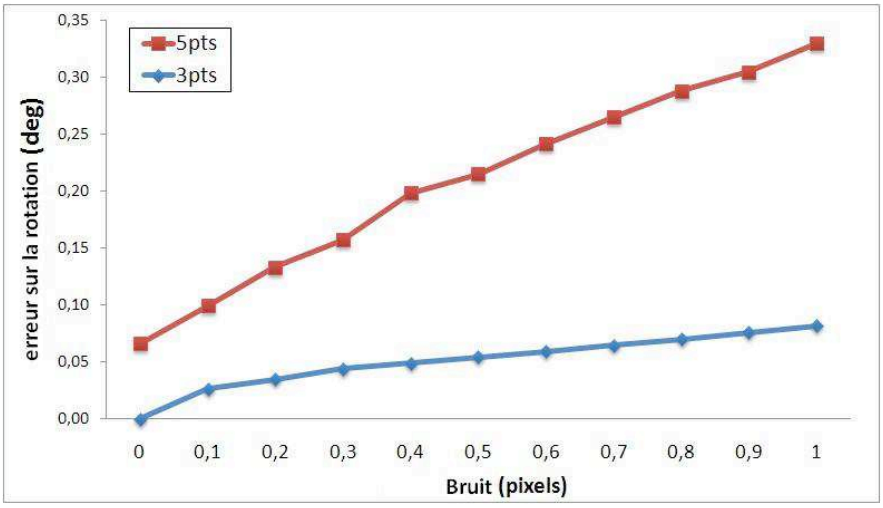



**Fig. 8.7.** Erreur sur la rotation (en degrés) en configuration plane (mouvement transverse)

.

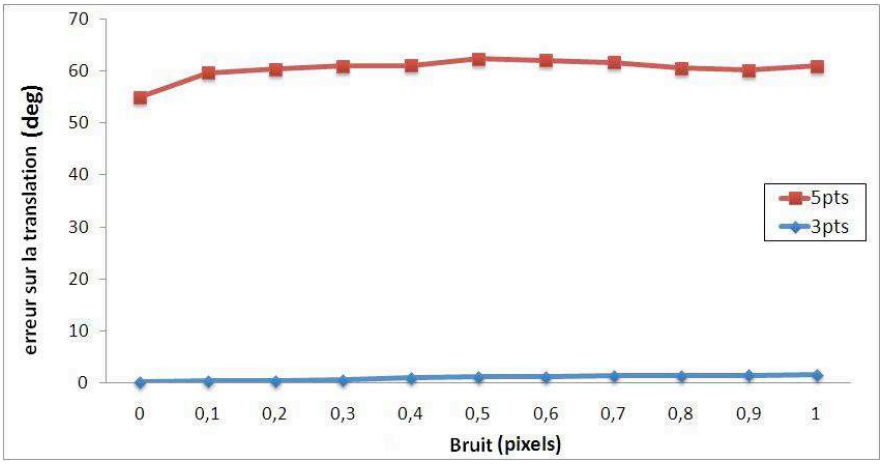



**Fig. 8.8.** Erreur sur l'orientation de la base (en degrés) en configuration plane (mouvement transverse)

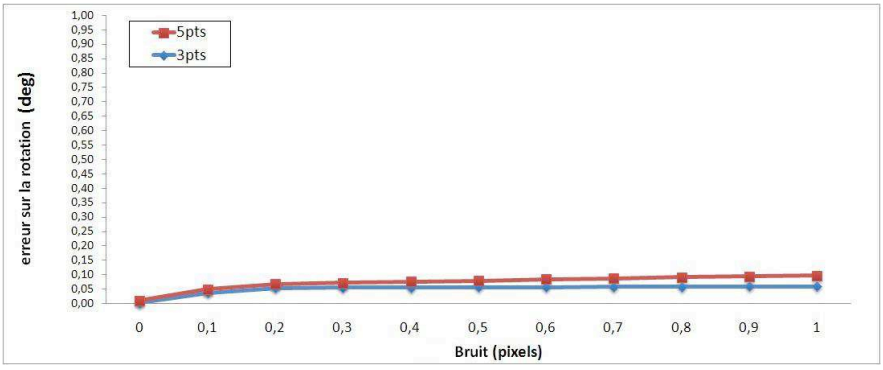



**Fig. 8.9.** Erreur sur la rotation (en degrés) en configuration plane (mouvement avant-arrière).

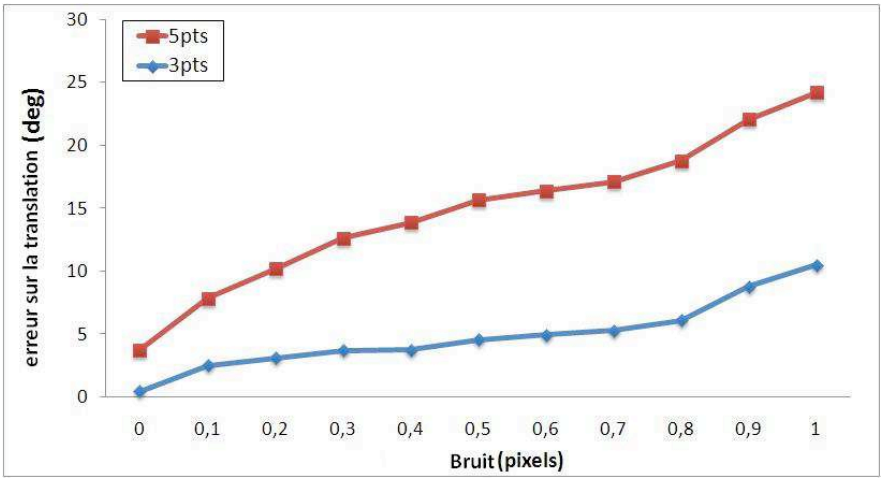
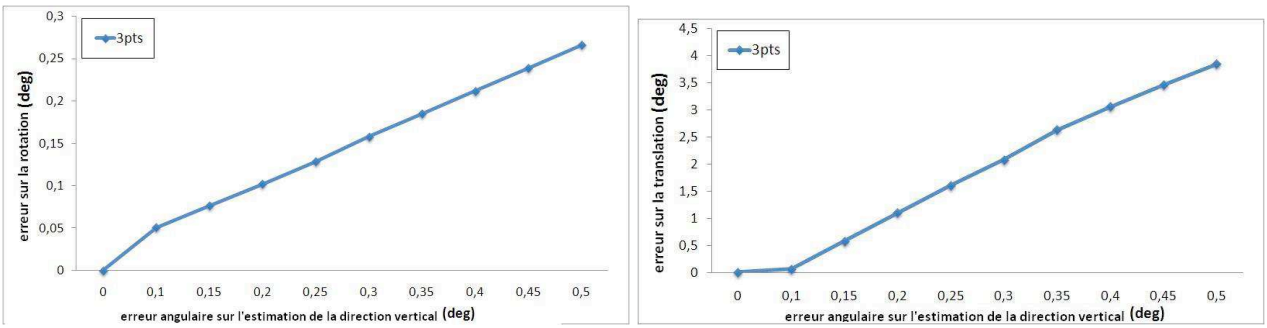



**Fig. 8.10.** Erreur sur l'orientation de la base (en degrés) en configuration plane (mouvement avant-arrière)

### Impact de la précision de la direction verticale sur l'estimation d'orientation relative

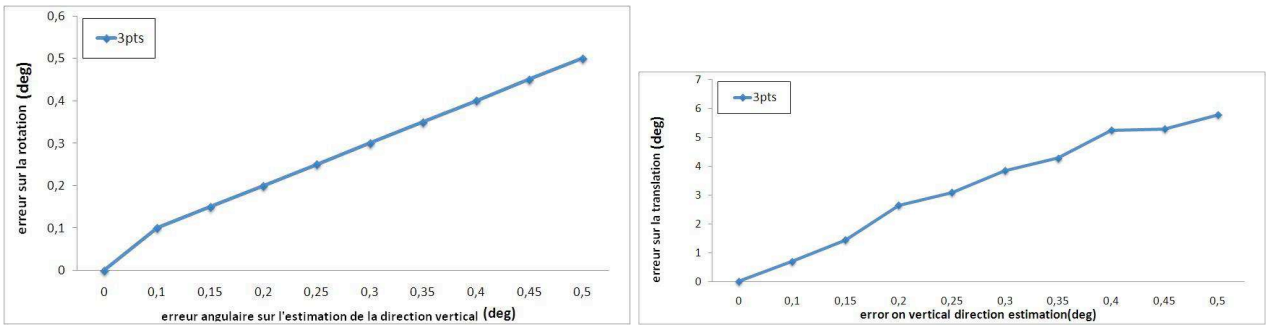
Nous avons introduit une erreur de 0 à 0.5 ° sur la précision angulaire de la direction verticale. Aujourd'hui par exemple, un dispositif inertiel à bas prix tel que le Xsens-MTi[1] donne une precison autour de 0.5 ° sur l'angle de rotation autour des axes X ou Z (la direction verticale étant selon l'axe Y). Bien sûr, il existe des centrales inertielles de grande précision, elles peuvent atteindre mieux que 0.01 ° sur les angles d'orientation quand elles sont associées correctement avec d'autres capteurs capables de compenser leur dérive (par exemple le GPS). En utilisant l'extraction automatique du point de fuite vertical, particulièrement dans une scène urbaine, nous obtenons une direction verticale très précise (bien mieux que 0.01 °), comme ce sera montré plus loin. Nous avons vérifié l'impact de cette précision sur la détermination de la rotation et de la base (Figures 8.11 et 8.12).




(a) (b)

**Fig. 8.11.** Impact de la précision géométrique sur la direction verticale sur l'estimation de a) la rotation (en degrés), et b) l'orientation de la base (en degrés) en mouvement transverse.

1. http://www.xsens.com/




(a) (b)

**Fig. 8.12.** Impact de la précision géométrique sur la direction verticale sur l'estimation de a) la rotation (en degrés), et b) l'orientation de la base (en degrés) en mouvement avant-arrière.

### 8.7.2 Exemple avec des données réelles

Afin de fournir un exemple numérique sur de vraies images, nous avons choisi de travailler sur la séquence de 9 images "entry-P10" de la base de données en ligne [Strecha *et al.*, 2008]. Dans cette base de données nous disposons de tous les paramètres intrinsèques et externes. En premier, nous avons extrait les points de fuite sur chaque image. Nous avons utilisé l'algorithme décrit dans 6.

Nous exprimons cette erreur en termes d'angles, et les résultats sont présentés dans la table 1. Comme on peut le voir, la détermination du point de fuite vertical est très précise et d'après les Figures 8.11 et 8.12 elle induit une erreur proche de zéro. Ensuite, nous avons

| Image | Erreur angulaire sur la direction verticale, en degrés |
|---|---|
| 0000 | 0.0026 |
| 0001 | 0.0066 |
| 0002 | 0.0016 |
| 0003 | 0.0014 |
| 0004 | 0.0009 |
| 0005 | 0.0011 |
| 0006 | 0.0014 |
| 0007 | 0.0050 |
| 0008 | 0.0024 |
| 0009 | 0.0022 |

**Tableau 8.1.** Résultats : Détection de la direction vertricale en utilisant le point de fuite.

calculé l'orientation relative pour 3 images consécutives (chaque fois, 2 couples d'images successives). Les points d'intérêt sont extraits en utilisant l'algorithme SIFT Lowe [2004].

Les résultats sont présentés dans la Figure 8.13. La valeur moyenne des erreurs angulaires sur la rotation s'élève à 0.82 degré. Pour l'estimation de la direction de la base, cette erreur monte à 1.33 degré. Ces résultats montrent clairement l'efficacité et la robustesse de la méthode.

| Error on the rotation (°) / Error on the translation (°) | 0000 | 0001 | 0002 | 0003 | 0004 | 0005 | 0006 | 0007 | 0008 | 0009 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0000  | — | 0.39 / 0.30 | 0.19 / 1.75 | | | | | | | |
| 0001  | | — | 0.30 / 1.94 | 0.35 / 1.61 | | | | | | |
| 0002  | | | — | 0.17 / 1.99 | 1.51 / 2.81 | | | | | |
| 0003  | | | | — | 0.61 / 0.001 | 0.58 / 1.81 | | | | |
| 0004  | | | | | — | 0.35 / 1.10 | 0.64 / 0.31 | | | |
| 0005  | | | | | | — | 0.54 / 0.65 | 1.16 / 1.79 | | |
| 0006  | | | | | | | — | 1.94 / 0.66 | 0.86 / 1.02 | |
| 0007  | | | | | | | | — | 0.74 / 0.89 | 2.21 / 1.21 |
| 0008  | | | | | | | | | — | 1.42 / 2.82 |
| 0009  | | | | | | | | | | — |

**Fig. 8.13.** Résultat sur la séquence "entry-P10". Chaque cellule contient l'erreur sur la rotation en degrés (en haut à gauche) et celle sur l'orientation de la base (en bas à droite).

### 8.7.3 Perfomances en temps de calcul

La résolution du système polynomial et l'extraction automatique du point de fuite ont été écrites en C ++. Avec un PC à 1.60 GHz, avec 2 Go de RAM, le temps de chaque résolution est approximativement de 2 $\mu s$, permettant des applications temps réel. Nous pouvons noter, sans surprise, que le processus de sélection qui utilise RanSac [Fischler et Bolles, 1981] parmi les points SIFT est considérablement plus rapide sur 3 points qu'avec l'algorithme de 5 points.

## 8.8 Conclusions

Aujourd'hui les appareils électroniques personnels comme les téléphones cellulaires, dont les prix sont de plus en plus bas, incluent des systèmes MEMS et inertiels en plus d'un appareil photo, permettant de fournir très facilement la direction de la verticale dans l'image. En outre, si besoin est, l'extraction automatique du point de fuite vertical par traitement d'image offre une alternative de très bonne précision, comme nous avons pu le voir précédemment : nous avons donc insisté sur l'avantage d'utiliser la connaissance de la direction verticale, et présenté un algorithme efficace pour résoudre le problème de l'orientation relative à partir de cette information. En plus d'une vitesse de calcul bien plus élevée, par comparaison avec la solution classique des 5 points, notre algorithme fournit une amélioration majeure : les scènes planes ne posent plus de problème d'orientation relative. Ce résultat est capital, particulièrement dans les scènes urbaines où le cas se présente très souvent, et il est dû à une formulation plus appropriée du problème, qui utilise de façon explicite les paramètres significatifs de l'orientation relative (paramètres de rotation et de translation). En outre, les autres avantages majeurs de cette nouvelle approche sont les suivants :

(i) L'extraction du point fuite correspondant à la direction verticale peut être faite de façon indépendante pour chaque image. Le calcul peut donc être effectué en programmation parallèle et par conséquent de manière rapide. Cette étape peut être remplacée par une mesure physique directe.

(ii) L'orientation utilisant le point de fuite vertical donne immédiatement 2 composantes de la rotation pour l'orientation absolue des images.

(iii) La simplification des équations de coplanarité permet une résolution directe en temps réel avec les bases de Gröbner.

(iv) Le processus de tri utilisant RanSac parmi les points SIFT est considérablement plus rapide sur 3 points que sur 5.

(v) Comme nous obtenons deux composantes de l'orientation absolue pour chaque image, la compensation par faisceaux devient considérablement plus stable que dans une situation classique.

# 9

# Calcul de l'orientation et de la position d'un ensemble d'images

## Sommaire



## 9.1 Introduction

Nous sommes désormais à la fin du processus : nous avons acquis des images, nous les avons orientées et localisées par couple, maintenant nous devons encore montrer comment on assemble de nombreux couples entre eux. Nous procédons donc à une mise en place approchée, afin de pouvoir ensuite effectuer une compensation par faisceaux. La compensation par faisceau, connue sous le nom de *bundle adjustement* en anglais, repose entièrement sur les équations de colinéarité. Or, comme on l'a vu précédemment, ces équations ne sont pas linéaires et nécessitent donc des valeurs initiales pour être linéarisées. Plus précisément, ceci permet la mise en géométrie d'un ensemble de N images dans le même référentiel.

Comme nous l'avons vu dans le chapitre 2.2.3, on peut donc dire que le relèvement est maintenant un problème dont la résolution est très bien connue. En théorie celle-ci ne nécessite que 3 points, mais en pratique il faut au moins un quatrième voire un cinquième point afin de trouver une solution unique, sans même parler du manque de fiabilité pratique d'un calcul basé sur le nombre minimal de points en cas de faute grossière. Nous avons donc essayé de trouver une alternative nouvelle.
Le relèvement permet la mise en référence d'une image par rapport à un modèle de points 3D, or on peut très bien mettre en référence des modèles 3D entre eux. Pour être plus précis, supposons que nous avons des points 3D issus du couple des images 1 et 2 (la référence du

système est le sommet de prise de vue de l'image 1). Si l'orientation relative est obtenue entre les images 2 et 3, nous aurons aussi des points 3D qui seront dans la référence de l'image 2. Afin de pouvoir mettre l'image 3 dans la première référence, il suffit de mettre les points 3D issus du couple 2 et 3 dans le référentiel du modèle issu du couple d'images 1 et 2. Pour illustrer ce point, ces étapes sont présentées dans la Figure 9.1.

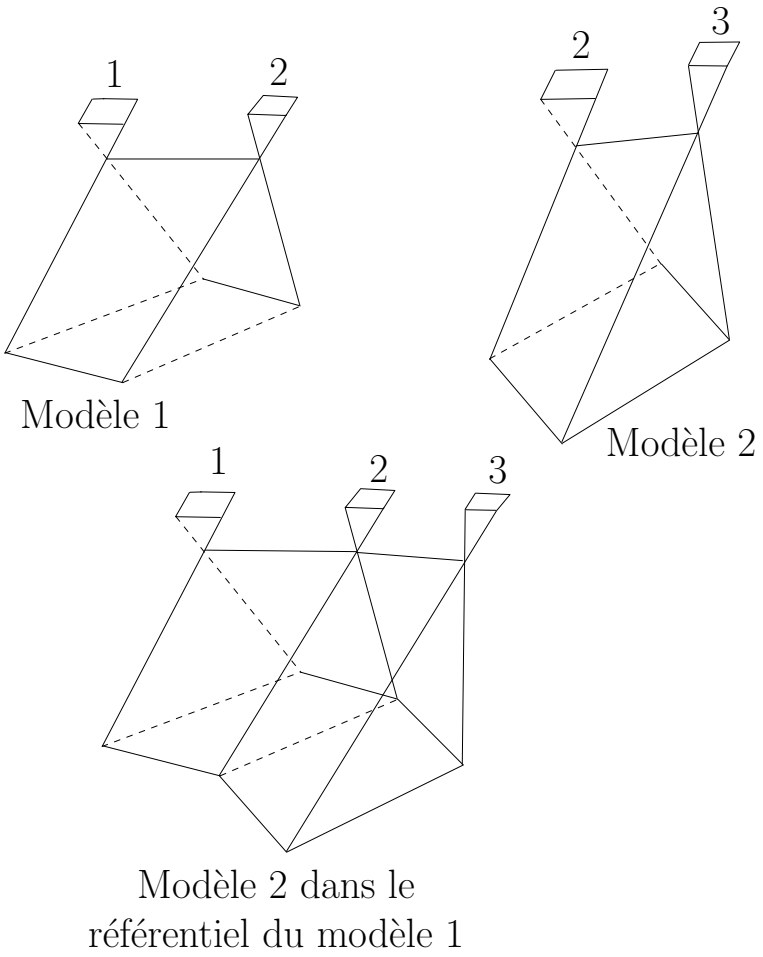



**Fig. 9.1.** Mise en référence d'un modèle par rapport à un autre.

Ce problème est l'équivalent du passage d'un système de référence à un autre, c'est donc une similitude dans l'espace à 3 dimensions.

Les équations de la similitude 3D sont celles utilisées pour l'orientation absolue (voir chapitre 2.2.3). Par ailleurs il est aussi possible de procéder à une compensation par modèle, au lieu d'une compensation par faisceaux, sous la dénomination technique de triangulation photogrammétrique : Compensation en bloc par modèles indépendants.

La transformation à 7 paramètres est très employée en géodésie [Awange et Grafarend, 2005], car elle permet le passage d'un système de référence à un autre. Elle transforme un

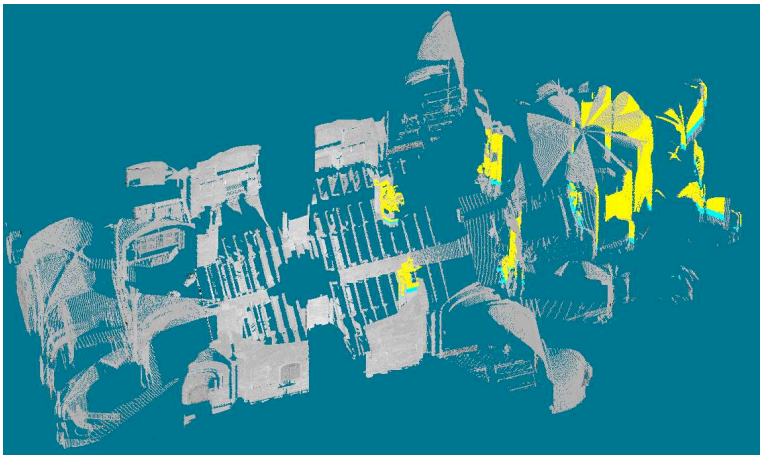
ensemble de points d'un système à un autre en utilisant une rotation, une translation et un facteur d'échelle.

En géodésie [Grafarend et Awange, 2003], [Vanicek et Krakiwsky, 1986], les angles de rotations étant très petits, une approximation est faite au niveau de la matrice de rotation. Elle donne lieu à ce qu'on appelle la transformation de Helmert. Dans un contexte de photogrammétrie, nous ne pouvons malheureusement pas faire cette simplification au niveau de la rotation, car ces angles peuvent être grands.

En photogrammétrie, nous allons employer cette transformation afin de connecter les différents modèles obtenus pour chaque couple. Le résultat sera un ensemble d'images, toutes dans la même référence.

C'est aussi le cas dans les relevés laser, afin de pouvoir mettre dans la même référence des nuages de points issus de stations différentes. Dans ce cas là, le facteur d'échelle est en général très proche de 1, cf.Figure 9.2.

a) Nuage de points acquis de puis la première station. b) Nuage de points asquis depuis la deuxième station.



c) Mise en réference de la deuxième station (jaune) par rapport à la première.

**Fig. 9.2.** Exemple de mise en référence des différents nuages laser présentés ci-dessus, en utilisant l'algorithme présenté ci-après (Monastier). crédit : ENSG

Dans ce chapitre un nouvel algorithme de résolution de la transformation à 7 paramètres est décrit. En effet avec l'aide des bases de Gröbner et d'une résolution polynomiale, nous pouvons trouver de manière directe, et sans besoin de linéariser, les valeurs de la translation, de la rotation et d'un facteur d'échelle.

Dans la suite, à l'aide de données simulées, nous confronterons l'algorithme à toutes sortes de configurations. Pour conclure, nous montrerons l'application de cet algorithme en photogrammétrie.

## 9.2 La méthode des 7 paramètres : calcul direct de la similitude 3D

### 9.2.1 Modélisation algébrique

La transformation à 7 paramètres contient, comme son nom l'indique, 7 inconnues, c'est-à-dire le facteur d'échelle $\lambda$, la rotation $R$ et la translation $Tx$, $Ty$, $Tz$. Comme on peut le voir sur la figure 9.3, une fois ces 7 paramètres obtenus, il est possible de pouvoir faire basculer tout le système 2 sur le système 1.

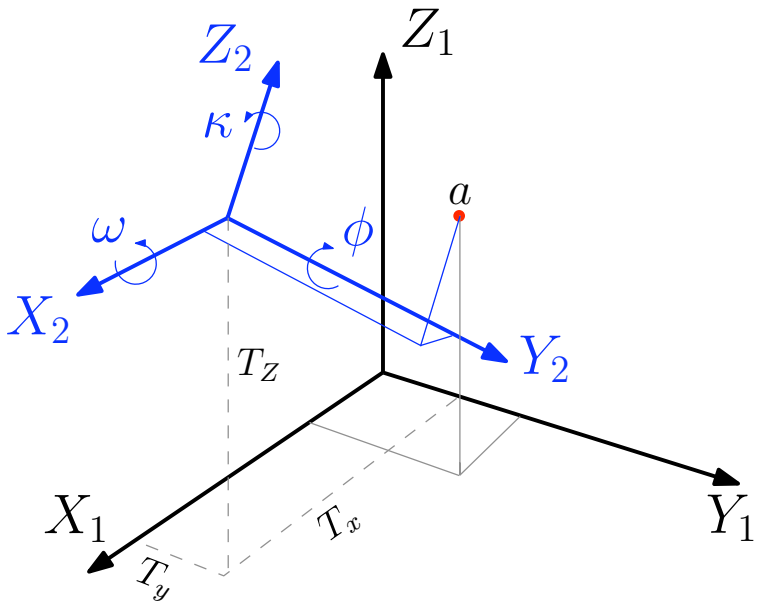



**Fig. 9.3.** Illustration 7 paramètres

$$\begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix}_1 = \lambda \; R(\omega, \phi, \kappa) \begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix}_2 + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{9.1}$$

où $R(\omega, \phi, \kappa)$ est la matrice de rotation (voir fig. 9.3).

Si on utilise la représentation de Cayley pour la matrice de rotation 7.4, nous aurons le système suivant [L. *et al.*, 2004] :

$$\begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix}_1 = \lambda\ (I-A)^{-1}(I+A) \begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix}_2 + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}. \tag{9.2}$$

*Rappel : A est une matrice antisymétrique.*
En multipliant les deux cotés de l'équation par $(I-A)$, nous obtenons :

$$(I-A) \begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix}_1 = \lambda\ (I+A) \begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix}_2 + (I-A) \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{9.3}$$

Afin de pouvoir résoudre les 7 paramètres, il est nécessaire d'avoir au minimum 3 points homologues dans les deux systèmes. En photogrammétrie, topographie et géodésie, il existe trois catégories de points d'appuis : des points dont on connait les coordonnés en X, Y, et Z, des points dont on connait seulement les coordonnées planimétriques en X et Y, et pour finir les points altimétriques dont on connait seulement les coordonnées en Z. Comme on peut le voir dans l'équation 9.1, chaque couple de points d'appuis connus en X, Y, Z , donne trois équations. Afin de pouvoir former un système de polynômes, il faudra sept équations, et donc disposer d'au moins un couple de points connu en X, Y, Z, ainsi qu'un couple connu en Z. En pratique on dispose presque toujours de points connus en X, Y et Z.

Pour trois points K, M et N exprimés dans les 2 systèmes nous fabriquons donc notre système à 7 équations et 7 inconnues :

$$\begin{aligned}
f_1 =& -Xk2 - cYk2 + bZk2 + \lambda Xk1 - \lambda cYk1 + \lambda bZk1 + T_x - cT_y + bT_z,\\
f_2 =& cXk2 - Yk2 - aZk2 + \lambda cXk1 + \lambda Yk1 - \lambda aZk1 + cT_x + T_y - aT_z,\\
f_3 =& -bXk2 + aYk2 - Zk2 - \lambda bXk1 + \lambda aYk1 + \lambda Zk1 - bT_x + aT_y + T_z,\\
f_4 =& -Xm2 - cYm2 + bZm2 + \lambda Xm1 - \lambda cYm1 + \lambda bZm1 + T_x - cT_y + bT_z,\\
f_5 =& cXm2 - Ym2 - aZm2 + \lambda cXm1 + \lambda Ym1 - \lambda aZm1 + cT_x + T_y - aT_z,\\
f_6 =& -bXm2 + aYm2 - Zm2 - \lambda bXm1 + \lambda aYm1 + \lambda Zm1 - bT_x + aT_y + T_z,\\
f_7 =& -bXn2 + aYn2 - Zn2 - \lambda bXn1 + \lambda aYn1 + \lambda Zn1 - bT_x + aT_y + T_z.
\end{aligned}$$

Les inconnues de ce système sont donc : $\lambda$, $a$, $b$, $c$, $T_x$, $T_y$ et $T_z$

### 9.2.2 Résolution à l'aide des bases de Gröbner

A la différence des équations de l'orientation relative vues précédemment, les paramètres de la translation et de la rotation ne sont pas entremêlés dans les équations des 7 paramètres. Ce qui simplifie de manière considérable les équations, car avec quelques combinaisons linéaires des équations les paramètres de la translation peuvent être supprimés. En effet nous pouvons envisager la combinaison suivante :

$$\begin{aligned} g_1 &= f_1 - f_4, \\ g_2 &= f_2 - f_5, \\ g_3 &= f_3 - f_7, \\ g_4 &= f_6 - f_7. \end{aligned}$$

Avec comme inconnues : $\lambda$, $a$, $b$ et $c$. Une fois ces inconnues obtenues il devient facile de calculer les paramètres de la translation, cf annexe D.

*Calcul de la matrice de Macaulay*

La démarche pour le calcul de la matrice de Macaulay $M$ une fois pour toute est exactement la même que celle définie dans la section 3.6. Dans ce cas-là, $M$ a une dimension de $24 \times 28$. Voici la liste des 24 polynômes qui ont été trouvés ainsi :

$$\begin{gathered} g_4, \lambda g_4, \lambda^2 g_4, \lambda^3 g_4, c g_4, \\ \lambda c g_4, \lambda^2 c g_4, \lambda^3 c g_4, g_3 \\ \lambda g_3, \lambda^2 g_3, \lambda^3 g_3, c g_3, \lambda c g_3 \\ \lambda^2 c g_3, \lambda^3 c g_3, g_2, \lambda g_2 \\ \lambda^2 g_2, \lambda^3 g_2, c g_2, \lambda c g_2 \\ \lambda^2 c g_2, \lambda^3 c g_2, g_1, \lambda, g_1 \\ \lambda^2 g_1, \lambda^3 g_1, c g_1, \lambda c, g_1 \\ \lambda^2 c g_1, \lambda^3 c g_1 \end{gathered}$$

Les détails sur cette matrice ainsi que l'obtention des paramètres inconnus $\lambda$, $a$, $b$, $c$, figurent dans l'annexe D.

## 9.3 Evaluation de la méthode des 7 paramètres

Dans cette partie nous évaluons avec des données synthétiques la précision de calcul avec l'aide de la méthode présentée dans ce chapitre. Pour cela il faut faire varier un bon nombre de paramètres, c'est-à-dire la rotation sur les trois axes, la translation en X, Y et Z, le facteur d'échelle. En plus de ces paramètres, il convient de bruiter les valeurs en entrée, avec des écarts-types réalistes. Cela nous permettra d'évaluer le comportement de l'algorithme en présence de bruit.

### 9.3.1 Configuration de la simulation

Comme il est sans intérêt de faire varier tous les paramètres en même temps, nous procédons en plusieurs étapes. Dans tous les cas les valeurs des coordonnées en entrée des deux systèmes sont bruitées avec des écarts types variant de 10 cm à 50 cm, avec un pas de 10 cm. Pour simplifier les écritures, les unités de longueur ne sont pas indiquées, puisqu'elles sont sans incidence sur les résultats. Ensuite plusieurs valeurs sur la rotation, translation et facteur d'échelle, sont traitées. A chaque fois, un seul de ces paramètres varie.

### 9.3.2 Calcul des points pour la simulation

Dans un cube de taille de 2000, 1000 points sont tirés de manière aléatoire. Ces 1000 points constitueront les points en entrée du premier système. Ensuite dans chaque configuration la rotation, translation et facteur d'échelle sont appliqués à ces 1000 points, afin d'obtenir un ensemble de coordonnées du deuxième système. Dans chaque configuration 500 tirages sont effectués. A titre d'exemple, pour une translation entre les deux de (5000, 1500, 500), un facteur d'échelle de 1.5 et une rotation avec comme angle de rotation autour de l'axe des X de 42 degrés, axe des y de 36 degrés et axe des z de 24 degrés, et un facteur d'échelle de 2.5, nous obtenons la figure 9.4 :

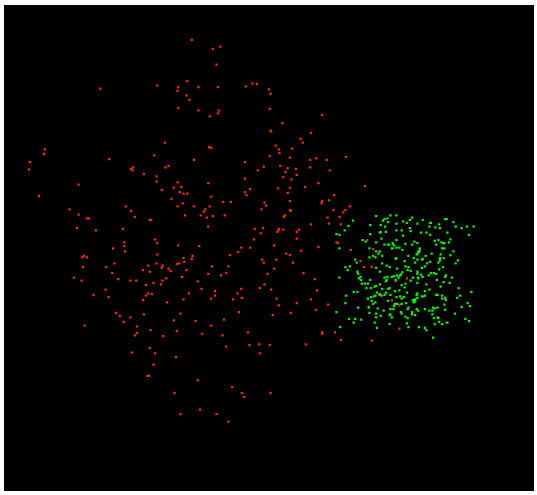


**Fig. 9.4.** En rouge les points dans le premier système. En vert les points fabriqués avec la configuration donnée.

Une fois que les deux jeux de coordonnées dans les deux systèmes sont fabriqués, nous bruitons les valeurs X, Y et Z de tous les points, avec des écarts type variant de 10 cm à 50 cm.

Touts les résultats de la simulation sont présentés dans l'annexe E.

## 9.4 Conclusions

Comme nous venons de le voir, le nouvel algorithme de résolution de la similitude 3D présenté a été testé dans différentes configurations. Cette méthode calcule les éléments de la similitude de manière directe avec l'aide des bases de Gröbner, et donc sans passer par des valeurs approchées.

Dans toutes ces simulations, des configurations très différentes ont été testées en présence de bruit. Comme on peut le constater, l'algorithme se comporte très bien dans toutes ces configurations. Sans représenter à proprement parler une preuve définitive, ce sont des indications fortes sur la fiabilité de cette méthode.

La méthode des 7 paramètres présentée dans ce chapitre peut être utilisée comme une alternative au relèvement classique. L'application de cette méthode s'étend tout naturellement bien au delà photogrammétrie et de la vision par ordinateur, comme par exemple dans le domaine de géodésie et topographie.

A ce stade, nous avons assemblé tous les éléments nécessaires au calcul de l'orientation et de la localisation d'un ensemble d'images.

# 10
# Conclusions et perspectives

La question majeure, qui a servi de fil conducteur tout au long de cette thèse, a été celle de l'orientation dans l'espace d'un ensemble d'images de milieux urbains, dans des situations où aucune information extérieure n'est disponible (données GPS, points d'appuis, etc..). Nous avons donc essayé d'apporter des réponses nouvelles à cette question.

Nous avons essayé d'améliorer les étapes classiques de mise en place d'un ensemble d'images dans l'espace avec les méthodes traditionnelles de photogrammétrie et de vision par ordinateur, en s'appuyant sur les points forts des travaux de chaque communauté.

Nous présentons dans la suite les contributions majeures de ce mémoire.

## Contributions

### La détection des points de fuite

Notre apport principal pour résoudre ce problème a été la mise au point de deux algorithmes entièrement automatiques, et fonctionnant quasiment en temps réel.

L'un de ces algorithmes ramène le problème de détection des points de fuite à la détection de cercles dans un nuage de points, et travaille entièrement dans l'espace image. L'autre algorithme projette tous les segments sur la sphère unité, et consiste à y détecter des plans.

Dans les deux cas, nous avons utilisé la méthode Ransac pour extraire de manière efficace les cercles ou les plans, moyennant quelques modifications bien adaptées. Initialement cette méthode est conçue pour n'extraire qu'un seul modèle dans un ensemble de données. La variante que nous avons mise au point dans cette étude permet l'extraction de plusieurs modèles dans un nuage de points, que ce soient des cercles ou bien des plans. Une de nos contributions importantes a été l'analyse complète du calcul de variance dans toute la chaîne de détection des points de fuite, depuis l'extraction de segments jusqu'au calcul de la localisation du point de fuite sur l'image. L'incertitude liée aux segments nous a permis de déterminer la tolérance applicable au Ransac. C'est grâce à ça que la détection des points de fuite est entièrement automatique, car cette tolérance est intrinsèque à l'image et directement liée à la qualité de détection des segments.

Enfin, les deux algorithmes présentés dans ce mémoire ne nécessitent pas la connaissance d'informations externes a priori, comme par exemple la calibration. De plus, ces deux méthodes ne font aucune estimation préalable du nombre de points de fuite qu'il faut détecter, différence appréciable par rapport à un grand nombre d'algorithmes publiés.

### L'orientation relative

#### L'orientation relative à l'aide de 5 point homologues

Le problème de l'orientation relative, bien que très ancien comme nous l'avons vu lorsque nous avons synthétisé l'état de l'art (chapitre 2), a été entièrement revisité dans ce mémoire. Avec l'aide des outils mathématiques puissants et récents que sont les bases de Gröbner, il a été possible de résoudre de manière directe les équations d'orientation relative avec l'aide de 5 points homologues (le strict minimum). L'avantage majeur par rapport aux solutions classiques en photogrammétrie est que la résolution de ce problème non-linéaire ne nécessite pas d'avoir des valeurs approchées pour la rotation et translation des images, valeurs parfois difficiles à obtenir.

Ce qui différencie l'approche présentée dans le chapitre 7 des méthodes de résolution classiques en vision par ordinateur, c'est qu'elle ne passe pas par un calcul de la matrice essentielle, mais plutôt par une résolution polynomiale directe dont les paramètres sont la rotation et la translation, en utilisant les bases de Gröbner.

Les résultats obtenus au cours des différentes simulations montrent que l'algorithme est peu sensible au bruit. Son autre point fort est la robustesse du calcul dans des cas ou tous les points sont coplanaires, y compris quand la base est parallèle à l'objet imagé. Cette robustesse est significativement meilleure que celle des résolutions basées sur la matrice essentielle.

#### Calcul de l'orientation relative à l'aide de la direction verticale et de 3 points homologues

En injectant la direction verticale comme information dans les équations de l'orientation relative basées sur la contrainte de coplanarité, nous avons proposé une nouvelle modélisation de celle-ci. Cette connaissance simplifie de manière considérable les équations de coplanarité et résout directement deux inconnues de la rotation. Au final 3 points homologues suffisent pour résoudre le reste des inconnues. La diminution de 5 points homologues à 3 entraine une accélération très conséquente dans le processus de Ransac. L'algorithme des 3 points est ainsi beaucoup plus rapide que l'algorithme des 5 points.

La connaissance de la direction verticale peut être donnée grâce à la détection des points de fuite à l'aide des algorithmes présentés dans la partie II. Cependant il est tout à fait envisageable d'utiliser une mesure physique directe comme par exemple une centrale inertielle intégrée (IMU) pour donner la direction verticale.

### L'orientation 3D d'un ensemble d'images

En supposant que l'on n'a aucune information externe (comme serait par exemple la connaissance de points d'appuis, ou bien de données GPS), nous avons proposé une méthode qui peut mettre en relation dans l'espace un ensemble d'image. Un couple d'images est pris comme référence, et ensuite à l'aide des équations de similitude 3D, nous calculons la rotation et la localisation de toutes les autres images. Nous avons résolu les équations de la similitude 3D, ici encore à l'aide des bases de Gröbner. L'algorithme mis au point se comporte de manière robuste dans toutes sortes de configurations. Cela a été largement démontré dans le chapitre 9.

## Perspectives et travaux futurs

Au cours de notre étude nous nous sommes efforcés de donner de nouvelles solutions à plusieurs problèmes de photogrammétrie et de vision par ordinateur, qui soient les plus générales possibles. Ces solutions ne sont pas liées à un cas particulier, ou à un instrument d'acquisition donné. La seule supposition faite tout au long de ce mémoire est que l'appareil photo est étalonné. Cet aspect très général permet donc une grande adaptabilité des travaux présentés aux différentes applications souhaitées.

Une des applications envisageables est de mettre en place une aide temps réel à l'acquisition d'images dans des campagnes photogrammétriques. Cette aide pourra créer le tableau d'assemblage au fur et à mesure des acquisitions. L'intérêt majeur de ce système est de garantir, sur le terrain même, qu'il n'existe pas de lacune dans l'acquisition, et que le recouvrement entre images est satisfaisant. Ceci est très important quand par exemple on procède à des missions loin de ses bases et que les données ne sont pas traitées sur place, et pour tous les autres cas où un retour sur terrain entraîne des surcoûts très élevés. Les algorithmes présentés dans ce mémoire donnent des solutions relatives à la position ainsi qu'à l'orientation des images entres elles, ce qui est très avantageux : en effet, avant même d'établir un réseau de points d'appui sur l'édifice, ou bien d'utiliser un lever GPS, on peut se trouver avec une configuration des caméras dans un référentiel arbitraire. La mise à l'échelle et le basculement de ce modèle peuvent se faire à tout moment, par rapport à un système de référence connu.

Un autre emploi de nos travaux est la prédiction d'une prise de vue optimale. Par exemple, imaginons que nous ayons pour objectif de relever les bâtiments de la rue Soufflot à Paris. Nous nous positionnons sur le trottoir d'en face afin de commencer nos acquisitions. Notre déplacement se fera transversalement afin d'acquérir des images avec des axes à peu près parallèles. Or, comme cette rue est une rue très passante de Paris (elle mène au Panthéon), notre démarche à toutes chances d'être perturbée par des piétons (voire par des véhicules). En plus de ça, cette rue forme un chantier long et délicat : un système expert peut vérifier à tout instant si les images acquises sont correctes géométriquement parlant, et ont assez de points homologues.

Les travaux développés peuvent également utilement être utilisés pour constituer un outil d'aide à la décision pour l'acquistion d'images en temps quasi réel. L'application consisterait

à aider un utilisateur à acquérir les images nécessaires pour répondre à des besoins spécifiques. Ces besoins peuvent être spécifiés en terme de recouvrement entre différentes images, précision géométrique de la reconstruction 3D escomptée (imposant des contraintes sur les ratios $\frac{B}{H}$), etc. Une insertion en temps réel des clichés au sein d'un tableau d'assemblage et une proposition de nouveau positionnement de la caméra avec son attitude pourra être proposée.

Les primitives extraites pour la constitution du tableau d'assemblage pourront également utilement servir à un calcul d'aérotriangulation permettant d'obtenir un positionnement plus précis des caméras. A l'issue, une incertitude pourra être associée à l'ensemble des partamètres estimés. Cette aérotriangulation pourra prendre en compte les points de fuites calculés associés à leur incertitude. Une étude plus précise sur l'incertitude des points de SIFT [Lowe, 2004] semble encore nécessaire pour mener à bien ce travail.

# Conclusions

Nous arrivons maintenant à la fin de cette étude. Nous avons étudié les méthodologies existantes en photogrammétrie et vision par ordinateur, pour l'orientation d'un bloc d'images dans l'espace. Cette analyse nous a conduits remplacer plusieurs maillons de la chaîne traditionnelle d'orientation d'images par de nouveaux composants plus performants.

Les améliorations ont porté sur :

- La mise en place de nouveaux algorithmes permettant de détecter de manière fiable les points de fuite. Ces deux méthodes sont basées sur des approches faisant largement usage d'outils mathématiques d'évaluation robuste, le Ransac.

- La résolution de manière directe d'un ensemble de problèmes classiques non-linéaires en photogrammétrie et vision par ordinateur. Par ailleurs ces nouveaux algorithmes permettent de trouver toutes les solutions aux équations posées.

- Une meilleure stabilité numérique. Les nouvelles méthodes d'orientation relatives (3 points et 5 points) sont stables pour l'ensemble des configurations testées dans le cadre de simulations ou dans le cadre d'orientation d'images réelles. Ces améliorations sont significatives compte tenu des comparaisons réalisées avec les méthodes de référence actuelles.

- Une meilleure stabilité numérique. Les nouvelles méthodes d'orientation relatives (3 points et 5 points) sont stables pour l'ensemble des configurations testées dans le cadre de simulations ou dans le cadre d'orientation d'images réelles. Ces améliorations sont significatives compte tenu des comparaisons réalisées avec les méthodes de référence actuelles.

- Une utilisation du nombre minimal de primitives pour former le modèle (5 points pour l'orientation relative ou 3 points et un point de fuite), 3 points pour l'algorithme des 7 paramètres. L'utilisation de configurations minimales est cruciale dans tous les algorithmes

d'estimation robuste utilisant des tirages aléatoires de type Ransac.

- Une utilisation du nombre minimal de primitives pour former le modèle (5 points pour l'orientation relative ou 3 points et un point de fuite), 3 points pour l'algorithme des 7 paramètres. L'utilisation de configurations minimales est cruciale dans tous les algorithmes d'estimation robuste utilisant des tirages aléatoires de type Ransac.

Par ailleurs, l'utilisation de l'algorithme des 7 paramètres dépasse très largement le cadre de la photogrammétrie et de la vision par ordinateur. En effet, ce type d'algorithme est d'un usage courant dans des domaines très variés : géodésie, topographie, etc.

Ces contributions innovantes ont été rendues possibles grâce aux progrès récents en géométrie algébrique et en particulier en prenant en compte les récentes avancées dans l'utilisation pratique des bases de Gröbner.

Pour montrer pleinement l'efficacité de ces nouvelles approches, celles ci pourront être intégrées dans une chaine de traitement permettant de produire des images orientées à partir d'images brutes.

# Bibliographie


Abdel-Aziz, Y. I. et Karara, H. M. (1971). Direct linear transformation from comparator coordinates into object space coordinates in closed range photogrammetry. *In ASPIUI Symposium on Close-Range Photogrammetry*, Urbana, IL.

Aguilera, D. et Gomez Lahoz, J.and Finat Codes, J. (2005). A new method for vanishing points detection in 3d reconstruction from a single view. *In ISPRS Commission V, WG V/2*, Mestre (Venice).

Almansa, A., Desolneux, A. et Vamech, S. (2003). Vanishing point detection without any a priori information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(4):502–507.

Alon, Y., Ferencz, A. et Shashua, A. (2006). Off-road path following using region classification and geometric projection constraints. *In CVPR '06 : Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 689–696, Washington, DC, USA. IEEE Computer Society.

Ameller, M., Quan, L. et Triggs, B. (2002). Le calcul de pose : de nouvelles méthodes matricielles. *In* AFRIF, éditeur : *14eme Congres Francophone de Reconnaissance des Formes et Intelligence Artificielle*.

Antone, M. et Teller, S. (2000). Automatic recovery of relative camera rotations for urban scenes. volume 02, pages 282–289, Los Alamitos, CA, USA. IEEE Computer Society.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R. et Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. of the ACM*, 45(6):89–923.

Awange, J, L. et Grafarend, E, W. (2005). *Solving Algebraic Computational Problems in Geodesy and Geoinformatics : The Answer to Modern Challenges*, volume XVII. Springer. 333p.

Barnard, S. T. (1983). Interpreting perspective images. *Artificial Intelligence*, 21:435–462.

Batista, A. L. (1452). *De la peinture*. Macula Dédale, Paris. Trad.J.L Scheffer,1992.

Batra, D., Nabbe, B. et Hebert, M. (2007). An alternative formulation for five point relative pose problem. pages 21–21.

Bauer, J., Klaus, A., Karner, K., Zach, C. et Schindler, K. (2002). Metropogis a city information system. *in Proceedings of IEEE Intern.Conf.on Image Processing*.

Bay, H., Ess, A., Tuytelaars, T. et Van Gool, L. (2008). Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359.

Beardsley, P. et Murray, D. (1992). Camera calibration using vanishing points. *In British Machine Vision Conference (BMVC92)*, pages 416–425.

Benallal, M. (2002). *Système de calibration de caméra : localisation de forme polyédrique par vision monoculaire*. Thèse de doctorat, Ecole des Mines de Paris.

Boulanger, K., Bouatouch, K. et Pattanaik, S. (2006). Atip : A tool for 3d navigation inside a single image with automatic camera calibration. *In EG UK conference on Theory and Practice of Computer Graphics*.

Brauer Burchardt, C. et Voss, K. (2000). Robust vanishing point determination in noisy images. *In International Conference on Pattern Recognition*, volume 1, pages 559–562.

Brauer-Burchardt, C. Voss, K. (2001). Facade reconstruction of destroyed buildings using historical photographs. *In Proceedings of the XVIII International Symposium CIPA 2001*, Potsdam (Germany).

Brillault-O'Mahoney, B. (1991). New method for vanishing point detection. *Computer Vision, Graphics, and Image Processing Image Understanding*, 54(2):289–300.

Brown, M. et Lowe, D. G. (2005). Unsupervised 3d object recognition and reconstruction in unordered datasets. *In 5th International Conference on 3D Imaging and Modelling, (3DIM2005)*, pages 56–63.

Buchberger, B. (1965). *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nuildimensionalen Polynomideal*. Thèse de doctorat, Universität Innsbruck.

Byröd, M., Josephson, K. et Åström, K. (2009). Fast and stable polynomial equation solving and its application to computer vision. *International Journal of Computer Vision*.

Cantoni, V., Lombardi, L., Porta, M. et Sicard, N. (2001). Vanishing point detection : representation analysis and new approaches. *In International Conference on Image Analysis and Processing*, pages 90–94.

Caprile, B. et Torre, V. (1990). Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4(2):127–140.

Cayley, A. (1846). Sur quelques propriétés des déterminants gauches. *Journal fur die Reine und Angewandte Mathematik (Crelle's Journal)*, (32):119–123.

Church, E. (1945). *Revised Geometry of the Aerial Photograph*. Bulletin Aerial Photogrammetry, Syracuse University.

Cipolla, R., Drummond, T. et Robertson, D. (1999). Camera calibration from vanishing points in image of architectural scenes. *In British Machine Vision Conference (BMVC99)*, page Posters/Exhibition/Demos.

Cole, L. et Barnes, N. (2008). Insect inspired three dimensional centring. *In Australasian Conference on Robotics and Automation (ACRA)*.

Collins, R. et Weiss, R. (1990). Vanishing point calculation as a statistical inference on the unit sphere. *In International Conference on Computer Vision (ICCV90)*, pages 400–403.

Cooper, M. A. R. (1987). *Control surveys in civil engineering*. Nichols Pub Co.

Cox, D., Little, J. et O'Shea, D. (1998). *Using algebraic geometry*, volume 185 de *Graduate Texts in Mathematics*. Springer-Verlag, New York.

Cox, D., Little, J. et O'Shea, D. (2004). *Using Algebraic Geometry*. Graduate texts in mathematics. Springer-Verlag New York-Berlin-Paris, 2 édition.

Cox, D., Little, J. et O'Shea, D. (2007a). *Ideals, varieties, and algorithms*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, third édition. An introduction to computational algebraic geometry and commutative algebra.

Cox, D., Little, J. et O'Shea, D. (2007b). *Ideals, varieties, and algorithms an introduction to computational algebraic geometry and commutative algebra*. Undergraduate texts in mathematics. Springer-Verlag New York-Berlin-Paris, 3 édition.

Criminisi, A., Reid, I. et Zisserman, A. (1999). Single view metrology. *International Journal of Computer Vision*, 40:123–148.

Demazure, M. (1988). Sur deux problemes de reconstruction. Rapport technique 882, INRIA.

DeMenthon, D. et Davis, L. (1992). Exact and approximate solutions of the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1100–1105.

Deriche, R. (1987). Using canny's criteria to derive an optimal edge detector recursively implemented. *Int. J. Computer Vision*, 2:15–20.

Deriche, R., Vaillant, R. et Faugeras, O. (1992). From noisy edges points to 3d reconstruction of a scene : A robust approach and its uncertainty analysis. *World Scientific Series in Machine Perception and Artificial Intelligence*, 2:71–79.

Fangi, G., Gagliardini, G. et Malinverni, E. (2001). Photointerpretation and small scale stereoplotting with digitally rectified photographs with geometrical constraints. *In Proceedings of the XVIII International Symposium CIPA 2001*, Potsdam (Germany).

Faugeras, O. (1993). *Three-Dimensional Computer Vision : A Geometric Viewpoint*. MIT Press.

Faugeras, O. D. et Maybank, S. (1990). Motion from point matches : multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246.

Faugère, J.-C. (1999). A new efficient algorithm for computing gröbner bases ($f_4$). *Journal of Pure and Applied Algebra*, 139(1-3):61–88.

Faugère, J., Merlet, J. et Rouillier, F. (2006). On solving the direct kinematics problem for parallel robots. Rapport technique 5923, INRIA.

Fischler, M. et Bolles, R. (1981). Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395.

Föstner, W. (1987). Reliability analysis of parameter estimation in linear models with application to mensuration problems in computer vision. *Computer Vision, Graphics and Image Processing*, 40(3):273–310.

Gallagher, A. (2005). Using vanishing points to correct camera rotation in images. *In The 2nd Canadian Conference on Computer and Robot Vision (CRV2005)*, pages 460– 467.

Gander, W., Golub, G. H. et Strebel, R. (1994). Fitting of circles and ellipses, least square solution. Rapport technique 1994TR-217, ETH Zurich, Department Informatik.

Georgiadis, C., Stefanidis, T., Gyftakis, S. et Agouris, P. (2005). Image orientation for interactive tours of virtually-modeled sites. *In 3D-ARCH'2005 : Virtual Reconstruction and Visualization of Complex Architectures*, Mestre-Venice, Italy.

Gonzalez-Vega, L. (1997). Implicitization of parametric curves and surfaces by using multidimensional newton formulae. *Journal of Symbolic Computation*, 23(2–3):137–152.

Grafarend, E. W. et Awange, J. L. (2003). Nonlinear analysis of the three-dimensional datum transformation [conformal group c7(3)]. *Journal of Geodesy*, 77(1):66–76.

Grammatikopoulos, L., Karras, G., Petsa, E. et Kalisperakis, I. (2006). *A unified approach for automatic camera calibration from vanishing points*, volume XXXVI de *5*. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.

Grayson, R. D. et Stillman, E. M. (1996). Macaulay 2, a software system for research in algebraic geometry. *Available at* `http ://wwww.math.uiuc.edu/Macaulay2`.

Grenier, L. (2006). Etudes complémentaires liée à la mise au point du photo-théodolite de l'ign : calibration et mesure de déformations. INSA Strasbourg. Projet de fin d'étude.

Greuel, G.-M., Pfister, G. et Schönemann, H. (2005). Singular 3.0. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern. `http ://www.singular.uni-kl.de`.

GRUNERT, J. (1841). Das pothenotische problem in erweiterter gestalt nebst bber seine anwendungen in der geodäsie. *Grunerts Archiv für Mathematik und Physik*, pages 238–248.

GRUSSENMEYER, P.and Al Khalil, O. (2002). Solutions for exterior orientation in photogrammetry, a review. *The photogrammetric record*, 17(100).

HARALICK, R., LEE, C., OTTENBERG, K. et NOLLE, M. (1994). Review and analysis of solutions of the three point perspective pose estimation problem. 13(3):331–356.

HARRIS, C. et STEPHENS, M. (1988). A combined corner and edge detection. *In Proceedings of The Fourth Alvey Vision Conference*, pages 147–151.

HARTLEY, R. I. et DANO, N. Y. (2000). Reconstruction from six-point sequences. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2480.

HARTLEY, R. I. et ZISSERMAN, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN : 0521540518, second édition.

HELMERT, F. R. (1872). *Die Ausgleichsrechnung nach der Methode der kleinsten Quadrate*. Teubner, Leipzig.

HEYDEN, A. et SPARR, G. (March 1999). Reconstruction from calibrated cameras-a new proof of the kruppa-demazure theorem. *Journal of Mathematical Imaging and Vision*, 10(20):123–142.

HORN, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America. A*, 4(4):629–642.

HOUGH, P. (1962). Method and means for recognizing complex patterns.

KAHL, F. et HARTLEY, R. (2008). Multiple-view geometry under the $L_\infty$-norm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1603–1617.

KANATANI, K. et ONODERA, Y. (1990). Camera calibration by computational projective geometry. *In MVA*, pages 363–366.

KARRAS, G. E. et PETSA, E. (2001). Metric information from single uncalibrated images. *In Proceedings of the XVII International Symposium CIPA 1999*, Olinda (Brazil).

KASSER, M. et EGELS, Y. (2001). *Photogrammetrie Numerique*. Edition Paris : Hermes Sciences Publications.

KENDER, J. (1979). Shape from texture : An aggregation transform that maps a class of textures into surface orientation. *In International Joint Conference on Artificial Intelligence (IJCAI79)*, pages 475–480.

KENNEDY, E. S. (1947). Al-kashi's plate of conjunctions. *Isis*, 38(1-2):56–59.

KONG, H., AUDIBERT, J. et JEAN PONCE, J. (2009). Vanishing point detection for road detection. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR2009)*.

KOSECKA, J. et ZHANG, W. (2002). Video compass. *In European Conference on Computer Vision*, page IV : 476 ff.

KOSECKA, J. et ZHANG, W. (2005). Extraction, matching and pose recovery based on dominant rectangular structures.

KRAUS, K. et WALDAEUSL, P. (1998). *Manuel de photogrammétrie, principes et procédés fondamentaux*. Hermès, Paris. Traduction de Grussenmeyer P. et Reis O., 407 pages.

KRONECKER, L. (1895-1931). *Werke*. Teubner, Leipzig.

KRUPPA, E. (1913). Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung. *Sitz.-Ber. Akad. Wiss., Wien, math. naturw. Abt. IIa.*, 122:1939–1948.

Kukelova, Z., Bujnak, M. et Pajdla, T. (2008). Polynomial eigenvalue solutions to the 5-pt and 6-pt relative pose problems.

L., A. J., Fukuda, Y. et Grafarend, E. W. (2004). Exact solution of the nonlinear 7-parameter datum transformation by groebner basis. *Bollettino di geodesia e scienze affini*, 63(2):117–127.

Lazard, D. (1983). Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. *In Computer algebra (London, 1983)*, volume 162 de *Lecture Notes in Comput. Sci.*, pages 146–156. Springer, Berlin.

Lee, S. C., Jung, S. K. et Nevatia, R. (2002). Automatic pose estimation of complex 3d building models. *In WACV*, pages 148–152. IEEE Computer Society.

Li, H. et Hartley, R. (2006). Five-point motion estimation made easy. pages I : 630–633.

Lobo, J. et Dias, J. (2003). Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1597–1608.

Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135.

Lourakis, M. et Argyros, A. (2004). The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Rapport technique 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Lutton, E. (1990). *Reconnaissance du point de prise de vue d'une photographie à partir d'un modèle de scène*. Thèse de doctorat, Ecole Télécom ParisTech ENST.

Lutton, E., Maitre, H. et Lopez-Krahe, J. (1994). Contribution to the determination of vanishing points using hough transform. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(4).

Ma, Y., Soatto, S., Kosecka, J. et Sastry, S. (2003). *An invitation to 3D vision, from images to models*. Springer Verlag.

Macek, K., Williams, B., Kolski, S. et Siegwart, R. (2004). A Lane Detection Vision Module for Driver Assistance. *In None*.

Magee, M. J. et Aggarwal, J. K. (1984). Determining vanishing points from perspective images. *Computer Vision, Graphics, and Image Processing*, 26(2):256–267.

Martinec, D. et Pajdla, T. (2007). Robust rotation and translation estimation in multiview reconstruction. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8.

Marzan, G. T. et Karara, H. M. (1975). A computer program for direct linear transformation solution of the colinearity condition, and some applications of it. *In* of Photogrammetry, A. S., éditeur : *Symposium on Close-Range Photogrammetric System*, Falls Church.

Maybank, S. (1992). *Theory of Reconstruction from Image Motion*. Springer-Verlag.

Merritt, E. (1949). Explicit three-point resection in space. *Photogrammetric Engineering*, 15(4):649–655.

Meyzonnette, J. (2003). *Optique géométrique et propagation*. Hermes Science Publications. 244 pages.

Mikolajczyk, K. et Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630.

Moffitt, F. et Mikhail, E. (1980). *Photogrammetry*. Cambridge : Harper& Row.

Muller, F. (1925). Direkte (exakte) lösung des einfachen ruckwârtseinschneidens im raume. *Teil, Allgemine Vermessungs Nachrichten*.

Nieto, M., Salgado, L., Jaureguizar, F. et Arrospide, J. (2008). Robust multiple lane road modeling based on perspective analysis. *In International Conference on Image Processing(ICIP08)*, pages 2396–2399.

Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777.

Nistér, D. (2004). A minimal solution to the generalised 3-point pose problem. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:560–567.

Ouchi, K., Keyser, J. et Rojas, J. M. (2003). The exact rational univariate representation and its application. Rapport technique 2003-11-1, Department of Computer Science, 3112 Texas A&M University, College Station, TX,77843-3112.

Patias, K. G. P. et Petsa, E. (1993). Experiences with rectification of non-metric digital images when ground control is not available. *In Proceedings of the XV International Symposium CIPA 2005*, Bucharest.

Philip, J. (1996). A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *Photogrammetric Record*, 15(88):589–599.

Philip, J. (1998). Critical point configurations of the 5-, 6-, 7-, and 8-point algorithms for relative orientations. Rapport technique, TRITA-MAT, KTH,Sweden.

Philippe, C. (1992). *La perspective en jeu- les dessous de l'image*. Découverte Gallimard-sciences, Paris.

Quan, L. et Lan, Z. (1999). Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:774–780.

Quan, L. et Mohr, R. (1989). Determining perspective structures using hierarchical hough transform. *Pattern Recognition Letters*, 9(4):279–286.

Radon, J. (1917). Uber die bestimmung von functionen durch ihre integralwerte langs gewisser mannigfaltigkeiten. *In Berichte Sachsische Academie der Wissenchaften*, volume 69, pages 262–267.

Rother, C. (2002). A new approach for vanishing point detection in architectural environments. *In Proceedings of the British Machine Vision Conference (BMVC)*, volume 20.

Rouillier, F. (1999). Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461.

Schindler, K. et Joachim, B. (2003). Towards feature-based building reconstruction from images. *In International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG2003)*.

Segvic, M., Schweighofer, G. et Pinz, A. (2007). Performance evaluation of the five-point relative pose with emphasis on planar scenes. *In Performance Evaluation for Computer Vision*, pages 33–40, Austria. Workshop of the Austrian Association for Pattern Recognition.

Semple, J. et Kneebone, G. (1952). *Algebraic projective geometry*. Oxford University Press.

Shaw, D. et Barnes, N. (2006). Perspective rectangle detection. *In Applications of Computer Vision, Workshop at the European Conference on Computer Vision (ECCV)*.

Shufelt, J. A. (1999). Performance evaluation and analysis of vanishing point detection techniques. *IEEE transactions PAMI*, 21(3):282–288.

Sidler, J. (2000). *Géométrie projective*. Dunod.

Simond, N. et Rives, P. (2004). Trajectography of an uncalibrated stereo rig in urban environments. *In IEEE RSJ/International conference on Intelligent Robot and System (IROS'04)*, pages 3381–3386, Sendai, Japan.

Slama, C. (1980). *Manual of Photogrammetry*. Book.

Snavely, N., Seitz, S. M. et Szeliski, R. (2008). Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.

Stentiford, F. W. M. (2006). Attention-based vanishing point detection. *In International Conference on Image Processing (ICIP2006)*.

Stewenius, H. (2005). Matlab code for solving the fivepoint problem. `http ://vis.uky.edu/~stewe/FIVEPOINT/`.

Stewénius, H., Engels, C. et Nistér, D. (2006). Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294.

Strecha, C., von Hansen, W., Van Gool, L., Fua, P. et Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. pages 1–8.

Sturm, R. (1869). Das problem der projektivität und seine anwendung auf die flächen zweiten grades. *In Math. Annal. 1*, pages 533–574.

Taillandier, F. (2004). *Reconstruction du bâti en milieu urbain : une approche multi-vues*. Thèse de doctorat, Ecole Polytechnique.

Teller, S., Antone, M., Bodnar, Z., Bosse, M., Coorg, S., Jethwa, M. et Master, N. (2001). Calibrated, registered images of an extended urban area. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:813.

Thompson, E. H. (1959). A rational algebraic formulation of the problem of relative orientation, photogrammetric record. *Photogrammetric Record*, 3(14):152–159.

Tomasi, C. et Kanade, T. (1992). Shape and motion from image streams under orthography : a factorization method. *Int. J. Comput. Vision*, 9(2):137–154.

Triggs, B. (1999). Camera pose and calibration from 4 or 5 known 3d points. *In In Proc. 7th Int. Conf. on Computer Vision*, pages 278–284. IEEE Computer Society Press.

Triggs, B. (2000). Routines for relative pose of two calibrated cameras from 5 points. Rapport technique, INRIA.

Tuytelaars, T., Gool, L. J. V., Proesmans, M. et Moons, T. (1998). A cascaded hough transform as an aid in aerial image interpretation. *In International Conference on Computer Vision*, pages 67–72.

Van den Heuvel, F. A. (1998). Vanishing point detection for architectural photogrammetry. *International Archives of Photogrammetry and Remote Sensing*, 32(5):652–659.

Vanicek, P. et Krakiwsky, E. (1986). *Geodesy : The Concepts*. North-Holland, Amsterdam, 2 édition.

Weiss, R. S., Nakatani, H. et Riseman, E. M. (1990). An error analysis for surface orientation from vanishing points. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(11):1179–1185.

Wilczkowiak, M., Sturm, P. et Boyer, E. (2005). Using geometric constraints through parallelepipeds for calibration and 3d modeling. 27(2):194–207.

Wolf, P. et Dewitt, B. (2000). *Elements of photogrammetry with applications in GIS*. Mcgraw-Hill.

Xie, W., Zhang, Z. et Zhang, J. (2004). *Multi-image Based Camera Calibration without Control Points*, volume 35 de *5*. International archives of Photogrammetry Remote Sensing and Spatial Information Sciences. p :36-41.

Zhang, W. et Kosecka, J. (2002). Efficient detection of vanishing points. *In IEEE International Conference on Robotics and Automation*.

# Résumé des publications de Mahzad Kalantari

**Articles de revues avec comité de lecture**

**M. Kalantari**, F. Jung, J.P. Guédon. Precise, automatic and fast method for vanishing point detection,The Photogrammetric Record, Vol. 24, No. 127., pp. 246-263, 2009.

**M. Kalantari**, F. Jung, N. Paparoditis, J.P. Guédon. Estimation de l'orientation relative : une approche directe basée sur la résolution de systèmes polynomiaux multivariables. Revue Française de Photogrammétrie et de Télédétection (RFPT), mai 2009. (Prix du meilleur article étudiant 2008).

**M. Kalantari**, M. Kasser. Photogrammétrie et vision par ordinateur, Revue XYZ, n°117, pp. 49-54, décembre 2008.

**M. Kalantari**, F. Jung. Estimation automatique de l'orientation relative en imagerie terrestre. Revue XYZ, n°114, pp. 41-46, mars 2008.

**M. Kalantari**, F. Jung. Détection entièrement automatique de points de fuite dans des scènes architecturales urbaines. Revue XYZ, n°107, pp. 41-46, juin 2006.

**Articles de conférence avec comité de lecture**

A. Hashemi, **M. Kalantari**. A Hybrid Algorithm for Solving 7 Parameters Transformation. 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Romania, 2009.

**M. Kalantari**, F. Jung, J.P. Guédon, N. Paparoditis. The Five Points Pose Problem : A New and Accurate Solution Adapted to any Geometric Configuration. IEEE Pacific-Rim Symposium on Image and Video Technology, (PSIVT) Tokyo, Japon, janvier 2009. Actes publiés dans "the Lecture Notes in Computer Science (LNCS)", Vol : 5414, Springer-Verlag.

**M. Kalantari**, F. Jung, J.P. Guédon, N. Paparoditis. Robust and Automatic Vanishing Points Detection with their Uncertainties from a Single Uncalibrated Image, by Planes Extraction on the Unit Sphere. XXI ISPRS Congress (Commission III, Part A), Pékin, Chine, juillet 2008.

**M. Kalantari**, F. Jung, J.P. Guédon, N. Paparoditis. Détection automatique des points de fuite et calcul de leur incertitude à l'aide de la géométrie projective. RFIA (Reconnaissance des Formes et Intelligence Artificielle), Amiens, France, janvier 2008.

**M. Kalantari**, M. Kasser. Implementation of a low-cost photogrammetric methodology for 3D modelling of ceramic fragments. CIPA, Athènes, Grèce, octobre 2007.

**M. Kalantari**, F. Jung, G. Moreau, J.P. Guédon. Détection entièrement automatique de points de fuite dans des scènes urbaines. COmpression et REprésentation des Signaux Audiovisuels (CORESA), Caen 2006.

# Partie IV

# Annexes

# A
# RanSac

L'algorithme RANSAC a été proposé par Fischler et Bolles [Fischler et Bolles, 1981]. Contrairement aux techniques de régression classiques qui consistent à traiter simultanément autant de points que possible, l'algorithme RANSAC repose sur l'estimation d'un modèle à partir du nombre minimal de $n$ points requis par le problème à traiter (ex : $n = 2$ points pour une droite, $n = 3$ points pour un cercle, $n = 5$ points pour l'estimation de l'orientation relative, etc.) tirés aléatoirement. Cette estimation est répétée plusieurs fois, le modèle retenu est celui qui a est retrouvé le plus grand nombre de fois lors de ces tirages qui impliquent la totalité des points. Un modèle retenu, le support, est défini comme étant le nombre de points dont la distance au modèle est inférieure à un certain seuil. En effet, l'idée intuitive de base consiste à dire que si un point faux a été utilisé pour l'estimation du modèle alors ce dernier n'aura pas un support d'importance comparable avec celui d'un modèle construit sur des points exacts.

Les valeurs des paramètres $t$ et $d$ doivent être déterminées à partir des exigences spécifiques relatives à l'application et l'ensemble de données, éventuellement sur la base d'évaluations expérimentales. Le paramètre $k$ (le nombre d'itérations), cependant, peut être déterminé à partir d'un résultat théorique.

Soit $P$ la probabilité qu'au moins un des n sous-ensembles tirés aléatoirement ne contienne pas un point faux. Supposons que $w$ est la probabilité de choisir un point bon à chaque tirage de point, c'est à dire : $w = \frac{nombrededonnesbonnes}{nombredepointsdedonnes}$.

En général, la valeur de $w$ est donnée à partir d'estimations, car on ne sait jamais de manière certaine le nombre de points faux, même si on en a toujours une idée.

Si nous supposons qu'on a besoin de $n$ points pour définir un modèle de manière indépendante, $w^n$ est la probabilité que les $n$ points soient tous justes, et $(1 - w)^n$, la probabilité qu'au moins un point parmi les $n$-points soit faux. Ce qui entraine l'estimation d'un modèle faux à partir de cet ensemble tiré au hasard.

On peut calculer le nombre de tirages qui garantit que, au moins dans un de ces tirages, un sous ensemble n'aura que des points bons, de la manière suivante : $k = \frac{\log(1-p)}{\log(1-w^n)}$.

**Algorithme 2** RanSac (Random Sample Consensus)

```
entrée :
données - un ensemble d'observations
modèle - le modèle qu'on veut estimer
n - le nombre minimum d'observations requis pour l'estimation du modèle
k - le nombre maximum d'itérations requis par l'algorithme
t - la valeur de tolérance pour savoir quand une observation appartient au modèle
d - le nombre d'observations requis pour être sur que le modèle correspond bien aux observations
sortie :
meilleur-modèle - les paramètres du modèle, qui correspondent le mieux aux données
meilleur-sousEnsemble - le groupe d'observations qui ont défini le modèle
meilleure-erreur - l'erreur du modèle relative aux données
iterations := 0
meilleur-modèle := null
meilleur-sousEnsemble := null
meilleur-erreur := infini
while iterations < k  do
   donnés-candidats := tirage de n données à partir des observations
   modèle-candidat := calcul des paramètres du modèle à partir des donnés-candidates
   sousEnsemble-candidat := donnés-candidates
   for chaque obervations hormis les donnés-candidates  do

      if les données ont une tolérance en dessous de t du modèle-candidat then
         ajout de ces points au meilleur-sousEnsemble

         if le nombre du meilleur-sousEnsemble>d  then
            meilleur-modèle-candidat := calcul des paramètres du modèle qui correspond au sousEnsemble-candidat

            cette-erreur := mesure l'erreur du modèle sur l'ensemble de ses points
         end if

         if cette-erreur < meilleure-erreur then
            meilleur-modèle := meilleur-modèle-candidat
            meilleur-sousEnsemble := sousEnsemble-candidat
            meilleur-erreur := cette-erreur
         end ifIncrémentation des itérations
      end if
   end for
end while
return  meilleur-modèle, meilleur-sousEnsemble, meilleure-erreur
```

Dans la figure A.1, on peut avoir le nombre de tirages nécessaires en fonction du nombre de paramètres à estimer (degré de liberté) et le pourcentage de points corrects [Hartley et Zisserman, 2004] :

Par exemple, on montre que pour estimer une droite ($n = 2$) passant par un ensemble de points avec 70% de points bons, il faudra 7 tirages (itérations).

| | Probabilité de points bons | | | | | | |
|---|---|---|---|---|---|---|---|
| n | 95% | 90% | 80% | 75% | 70% | 60% | 50% |
| 2 | 2 | 3 | 5 | 6 | 7 | 11 | 17 |
| 3 | 3 | 4 | 7 | 9 | 11 | 19 | 35 |
| 4 | 3 | 5 | 9 | 13 | 17 | 34 | 72 |
| 5 | 4 | 6 | 12 | 17 | 26 | 57 | 146 |
| 6 | 4 | 7 | 16 | 24 | 37 | 97 | 293 |
| 7 | 4 | 8 | 20 | 33 | 54 | 163 | 588 |
| 8 | 5 | 9 | 26 | 44 | 78 | 272 | 1177 |

**Fig. A.1.** Tableau montrant la détermination du nombre de tirages en fonction du nombre de paramètres à estimer ($n$) et du pourcentage de points bons.

# B

# Estimation directe de l'orientation relative à l'aide de 5 points

Dans cette annexe nous décrivons comment avec l'aide des librairies Salsa et OPenMaple il est possible d'appeler, directement depuis un programme c++, Fortran et Java, certaines fonctions mathématiques.

Nous donnons les éléments nécessaires afin de résoudre les équations de l'orientation relative à partir de 5 points, comme décrit dans le chapitre 7.

La première chose à faire quand on utilise OpenMaple c'est de définir un noyau, avec la commande MKernelVector kv , kv étant le nom de variable du noyau.

Pile-Image-Ransac contient les coordonnés des 5 points homologues issus du Ransac.

```
MKernelVector kv;
if( ( kv = StartMaple(argc, argv, &cb, NULL, NULL, err) ) == NULL )  return( 1 );

ALGEB gb-drl, test-dim, rr,sols;

ALGEB l = MapleListAlloc(kv,3);

EvalMapleStatement(kv,"restart :with(Gb) :with(FGb) :advance(FGB) :with(RS) :");

//***************** Definiton des relations **************************
ALGEB r1= EvalMapleStatement(kv,"r1 := (xa1*(-4*Tz*v-2*Tz*l*u-4*Ty*u+2*Ty*l*v)+ya1*(4*Tz+Tz*l^2-
Tz*u^2-Tz*v^2+4*Tx*u-2*Tx*l*v)+za1*(-4*Ty-Ty*l^2+Ty*u^2+Ty*v^2+4*Tx*v+2*Tx*l*u))*xa2+(xa1*(-4*
Tz+Tz*l^2-Tz*u^2+Tz*v^2+4*Ty*l+2*Ty*u*v)+ya1*(-4*Tz*v+2*Tz*l*u-4*Tx*l-2*Tx*u*v)+za1*(4*Ty*v-
2*Ty*l*u+4*Tx-Tx*l^2+Tx*u^2-Tx*v^2))*ya2+(xa1*(4*Tz*l-2*Tz*u*v+4*Ty-Ty*l^2-Ty*u^2+Ty*v^2)+ya1*
(4*Tz*u+2*Tz*l*v-4*Tx+Tx*l^2+Tx*u^2-Tx*v^2)+za1*(-4*Ty*u-2*Ty*l*v-4*Tx*l+2*Tx*u*v))*za2 :");
ALGEB r2 = EvalMapleStatement(kv,"r2 := (xb1*(-4*Tz*v-2*Tz*l*u-4*Ty*u+2*Ty*l*v)+yb1*(4*Tz+Tz*
l^2-Tz*u^2-Tz*v^2+4*Tx*u-2*Tx*l*v)+zb1*(-4*Ty-Ty*l^2+Ty*u^2+Ty*v^2+4*Tx*v+2*Tx*l*u))*xb2+(xb1*
(-4*Tz+Tz*l^2-Tz*u^2+Tz*v^2+4*Ty*l+2*Ty*u*v)+yb1*(-4*Tz*v+2*Tz*l*u-4*Tx*l-2*Tx*u*v)+zb1*(4*
Ty*v-2*Ty*l*u+4*Tx-Tx*l^2+Tx*u^2-Tx*v^2))*yb2+(xb1*(4*Tz*l-2*Tz*u*v+4*Ty-Ty*l^2-Ty*u^2+Ty*v^2)+
yb1*(4*Tz*u+2*Tz*l*v-4*Tx+Tx*l^2+Tx*u^2-Tx*v^2)+zb1*(-4*Ty*u-2*Ty*l*v-4*Tx*l+2*Tx*u*v))*zb2 :");
ALGEB r3 = EvalMapleStatement(kv,"r3 := (xc1*(-4*Tz*v-2*Tz*l*u-4*Ty*u+2*Ty*l*v)+yc1*(4*Tz+Tz*
l^2-Tz*u^2-Tz*v^2+4*Tx*u-2*Tx*l*v)+zc1*(-4*Ty-Ty*l^2+Ty*u^2+Ty*v^2+4*Tx*v+2*Tx*l*u))*xc2+(xc1*
(-4*Tz+Tz*l^2-Tz*u^2+Tz*v^2+4*Ty*l+2*Ty*u*v)+yc1*(-4*Tz*v+2*Tz*l*u-4*Tx*l-2*Tx*u*v)+zc1*(4*
Ty*v-2*Ty*l*u+4*Tx-Tx*l^2+Tx*u^2-Tx*v^2))*yc2+(xc1*(4*Tz*l-2*Tz*u*v+4*Ty-Ty*l^2-Ty*u^2+Ty*v^2)+
yc1*(4*Tz*u+2*Tz*l*v-4*Tx+Tx*l^2+Tx*u^2-Tx*v^2)+zc1*(-4*Ty*u-2*Ty*l*v-4*Tx*l+2*Tx*u*v))*zc2 :");
ALGEB r4 = EvalMapleStatement(kv,"r4 := (xd1*(-4*Tz*v-2*Tz*l*u-4*Ty*u+2*Ty*l*v)+yd1*(4*Tz+Tz*
l^2-Tz*u^2-Tz*v^2+4*Tx*u-2*Tx*l*v)+zd1*(-4*Ty-Ty*l^2+Ty*u^2+Ty*v^2+4*Tx*v+2*Tx*l*u))*xd2+(xd1*
(-4*Tz+Tz*l^2-Tz*u^2+Tz*v^2+4*Ty*l+2*Ty*u*v)+yd1*(-4*Tz*v+2*Tz*l*u-4*Tx*l-2*Tx*u*v)+zd1*(4*
```

$Ty*v-2*Ty*l*u+4*Tx-Tx*l^2+Tx*u^2-Tx*v^2))*yd2+(xd1*(4*Tz*l-2*Tz*u*v+4*Ty-Ty*l^2-Ty*u^2+Ty*v^2)+yd1*(4*Tz*u+2*Tz*l*v-4*Tx+Tx*l^2+Tx*u^2-Tx*v^2)+zd1*(-4*Ty*u-2*Ty*l*v-4*Tx*l+2*Tx*u*v))*zd2$ :") ;
ALGEB r5 = EvalMapleStatement(kv,"$r5 := (xe1*(-4*Tz*v-2*Tz*l*u-4*Ty*u+2*Ty*l*v)+ye1*(4*Tz+Tz*l^2-Tz*u^2-Tz*v^2+4*Tx*u-2*Tx*l*v)+ze1*(-4*Ty-Ty*l^2+Ty*u^2+Ty*v^2+4*Tx*v+2*Tx*l*u))*xe2+(xe1*(-4*Tz+Tz*l^2-Tz*u^2+Tz*v^2+4*Ty*l+2*Ty*u*v)+ye1*(-4*Tz*v+2*Tz*l*u-4*Tx*l-2*Tx*u*v)+ze1*(4*Ty*v-2*Ty*l*u+4*Tx-Tx*l^2+Tx*u^2-Tx*v^2))*ye2+(xe1*(4*Tz*l-2*Tz*u*v+4*Ty-Ty*l^2-Ty*u^2+Ty*v^2)+ye1*(4*Tz*u+2*Tz*l*v-4*Tx+Tx*l^2+Tx*u^2-Tx*v^2)+ze1*(-4*Ty*u-2*Ty*l*v-4*Tx*l+2*Tx*u*v))*ze2$ :") ;
ALGEB r6 = EvalMapleStatement(kv,"$r6 := Tx^2 + Ty^2 + Tz^2 - 1$ :") ;

//********** Commande sur maple pour l'initialisation **********
MapleAssign(kv,ToMapleName(kv,"xa1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(0).x)) ;
MapleAssign(kv,ToMapleName(kv,"ya1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(0).y)) ;
MapleAssign(kv,ToMapleName(kv,"xb1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(1).x)) ;
MapleAssign(kv,ToMapleName(kv,"yb1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(1).y)) ;
MapleAssign(kv,ToMapleName(kv,"xc1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(2).x)) ;
MapleAssign(kv,ToMapleName(kv,"yc1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(2).y)) ;
MapleAssign(kv,ToMapleName(kv,"xd1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(3).x)) ;
MapleAssign(kv,ToMapleName(kv,"yd1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(3).y)) ;
MapleAssign(kv,ToMapleName(kv,"xe1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(4).x)) ;
MapleAssign(kv,ToMapleName(kv,"ye1",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(0)(4).y)) ;

MapleAssign(kv,ToMapleName(kv,"za1",TRUE),ToMapleInteger(kv,focale)) ;
MapleAssign(kv,ToMapleName(kv,"zb1",TRUE),ToMapleInteger(kv,focale)) ;
MapleAssign(kv,ToMapleName(kv,"zc1",TRUE),ToMapleInteger(kv,focale)) ;
MapleAssign(kv,ToMapleName(kv,"zd1",TRUE),ToMapleInteger(kv,focale)) ;
MapleAssign(kv,ToMapleName(kv,"ze1",TRUE),ToMapleInteger(kv,focale)) ;

MapleAssign(kv,ToMapleName(kv,"xa2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(0).x)) ;
MapleAssign(kv,ToMapleName(kv,"ya2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(0).y)) ;
MapleAssign(kv,ToMapleName(kv,"xb2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(1).x)) ;
MapleAssign(kv,ToMapleName(kv,"yb2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(1).y)) ;
MapleAssign(kv,ToMapleName(kv,"xc2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(2).x)) ;
MapleAssign(kv,ToMapleName(kv,"yc2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(2).y)) ;
MapleAssign(kv,ToMapleName(kv,"xd2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(3).x)) ;
MapleAssign(kv,ToMapleName(kv,"yd2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(3).y)) ;
MapleAssign(kv,ToMapleName(kv,"xe2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(4).x)) ;
MapleAssign(kv,ToMapleName(kv,"ye2",TRUE),ToMapleInteger(kv,Pile-Image-Ransac(1)(4).y)) ;

MapleAssign(kv,ToMapleName(kv,"za2",TRUE),ToMapleInteger(kv,focale)) ;
MapleAssign(kv,ToMapleName(kv,"zb2",TRUE),ToMapleInteger(kv,focale)) ;
MapleAssign(kv,ToMapleName(kv,"zc2",TRUE),ToMapleInteger(kv,focale)) ;
MapleAssign(kv,ToMapleName(kv,"zd2",TRUE),ToMapleInteger(kv,focale)) ;
MapleAssign(kv,ToMapleName(kv,"ze2",TRUE),ToMapleInteger(kv,focale)) ;

r1 = MapleEval(kv,r1) ;
r2 = MapleEval(kv,r2) ;
r3 = MapleEval(kv,r3) ;
r4 = MapleEval(kv,r4) ;
r5 = MapleEval(kv,r5) ;
r6 = MapleEval(kv,r6) ;

ALGEB li = MapleListAlloc(kv,6) ;

MapleListAssign(kv,li,1,r1) ;
MapleListAssign(kv,li,2,r2) ;
MapleListAssign(kv,li,3,r3) ;
MapleListAssign(kv,li,4,r4) ;
MapleListAssign(kv,li,5,r5) ;

MapleListAssign(kv,li,6,r6) ;

ALGEB Rela= EvalMapleStatement(kv,"Rela :=[r1,r2,r3,r4,r5,r6] :") ;
gb-drl= EvalMapleStatement(kv,"gb-drl :=gbasis(Rela,DRL([l,u,v,Tx,Ty,Tz])) :") ;
test-dim = EvalMapleStatement(kv,"dimension(gb-drl) :") ;
int dimension = MapleToInteger8(kv,test-dim) ;
rr= EvalMapleStatement(kv,"rr := $rs_rur$(gb-drl,verbose=1,output=poly) :") ;
sols=EvalMapleStatement(kv,"sols :=$rs_isolate$(rr,vars=vars,order=R̈UR;scalars=$floating_point$) ;") ;
ALGEB nopsS = EvalMapleStatement(kv,"nbSols :=nops(sols) ;") ;

# B.1 Conclusions

Dans cette annexe, nous avons décrit les équations et la syntaxe nécessaires afin de résoudre les équations d'orientation relative à l'aide des 5 points.

# C

# Orientation relative à partir de 3 points homologues et de la direction verticale

Dans cette annexe nous décrivons la matrice de Macaulay, que nous avons fabriquée afin de résoudre les équations décrites dans le chapitre 8. Cette matrice que nous appelons M, est une matrice qui a 71 lignes et 85 colonnes. En effectuant une élimination du type Gauss-Jordan sur cette matrice, nous obtenons les bases de Gröbner. Dans la suite, nous donnons les matrices d'action (ANGLE, TRANSX, TRANSY, TRANSZ) pour chacune des inconnues. Le calcul des valeurs propres de cette matrice donne la solution pour chaque inconnue.

Matrice M(71,85)

M(0,57) =zc2 * yc1 ;
M(0,58) =-2 * xc2 * yc1 ;
M(0,59) =yc2 * zc1 - zc2 * yc1 ;
M(0,65) =xc2 * zc1 - zc2 * xc1 ;
M(0,66) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(0,67) =-xc2 * zc1 + zc2 * xc1 ;
M(0,73) =-xc2 * yc1 ;
M(0,74) =-2 * zc2 * yc1 ;
M(0,75) =xc2 * yc1 - yc2 * xc1 ;
M(1,56) =zc2 * yc1 ;
M(1,57) =-2 * xc2 * yc1 ;
M(1,58) =yc2 * zc1 - zc2 * yc1 ;
M(1,64) =xc2 * zc1 - zc2 * xc1 ;
M(1,65) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(1,66) =-xc2 * zc1 + zc2 * xc1 ;
M(1,72) =-xc2 * yc1 ;
M(1,73) =-2 * zc2 * yc1 ;
M(1,74) =xc2 * yc1 - yc2 * xc1 ;
M(2,55) =zc2 * yc1 ;
M(2,56) =-2 * xc2 * yc1 ;
M(2,57) =yc2 * zc1 - zc2 * yc1 ;
M(2,63) =xc2 * zc1 - zc2 * xc1 ;
M(2,64) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(2,65) =-xc2 * zc1 + zc2 * xc1 ;
M(2,71) =-xc2 * yc1 ;
M(2,72) =-2 * zc2 * yc1 ;
M(2,73) =xc2 * yc1 - yc2 * xc1 ;
M(3,54) =zc2 * yc1 ;
M(3,55) =-2 * xc2 * yc1 ;
M(3,56) =yc2 * zc1 - zc2 * yc1 ;
M(3,62) =xc2 * zc1 - zc2 * xc1 ;
M(3,63) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(3,64) =-xc2 * zc1 + zc2 * xc1 ;
M(5,22) =-xc2 * zc1 + zc2 * xc1 ;
M(5,38) =-xc2 * yc1 ;
M(5,39) =-2 * zc2 * yc1 ;
M(5,40) =xc2 * yc1 - yc2 * xc1 ;
M(6,10) =zc2 * yc1 ;
M(6,11) =-2 * xc2 * yc1 ;
M(6,12) =yc2 * zc1 - zc2 * yc1 ;
M(6,19) =xc2 * zc1 - zc2 * xc1 ;
M(6,20) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(6,21) =-xc2 * zc1 + zc2 * xc1 ;
M(6,37) =-xc2 * yc1 ;
M(6,38) =-2 * zc2 * yc1 ;
M(6,39) =xc2 * yc1 - yc2 * xc1 ;
M(7,9) = zc2 * yc1 ;
M(7,10) =-2 * xc2 * yc1 ;
M(7,11) =yc2 * zc1 - zc2 * yc1 ;
M(7,18) =xc2 * zc1 - zc2 * xc1 ;
M(7,19) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(7,20) =-xc2 * zc1 + zc2 * xc1 ;
M(7,36) =-xc2 * yc1 ;
M(7,37) =-2 * zc2 * yc1 ;
M(7,38) =xc2 * yc1 - yc2 * xc1 ;
M(8,6) = zc2 * yc1 ;
M(8,7) = -2 * xc2 * yc1 ;
M(8,8) = yc2 * zc1 - zc2 * yc1 ;
M(8,15) =xc2 * zc1 - zc2 * xc1 ;
M(8,16) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(8,17) =-xc2 * zc1 + zc2 * xc1 ;
M(8,33) =-xc2 * yc1 ;
M(8,34) =-2 * zc2 * yc1 ;
M(8,35) =xc2 * yc1 - yc2 * xc1 ;
M(9,5) = zc2 * yc1 ;
M(9,6) = -2 * xc2 * yc1 ;

```
M(3,70) =-xc2 * yc1 ;
M(3,71) =-2 * zc2 * yc1 ;
M(3,72) =xc2 * yc1 - yc2 * xc1 ;
M(4,12) =zc2 * yc1 ;
M(4,13) =-2 * xc2 * yc1 ;
M(4,14) =yc2 * zc1 - zc2 * yc1 ;
M(4,21) =xc2 * zc1 - zc2 * xc1 ;
M(4,22) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(4,23) =-xc2 * zc1 + zc2 * xc1 ;
M(4,39) =-xc2 * yc1 ;
M(4,40) =-2 * zc2 * yc1 ;
M(4,41) =xc2 * yc1 - yc2 * xc1 ;
M(5,11) =zc2 * yc1 ;
M(5,12) =-2 * xc2 * yc1 ;
M(5,13) =yc2 * zc1 - zc2 * yc1 ;
M(5,20) =xc2 * zc1 - zc2 * xc1 ;
M(5,21) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(11,4) =-2 * xc2 * yc1 ;
M(11,5) =yc2 * zc1 - zc2 * yc1 ;
M(11,12) =xc2 * zc1 - zc2 * xc1 ;
M(11,13) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(11,14) =-xc2 * zc1 + zc2 * xc1 ;
M(11,30) =-xc2 * yc1 ;
M(11,31) =-2 * zc2 * yc1 ;
M(11,32) =xc2 * yc1 - yc2 * xc1 ;
M(12,2) =zc2 * yc1 ;
M(12,3) =-2 * xc2 * yc1 ;
M(12,4) =yc2 * zc1 - zc2 * yc1 ;
M(12,11) =xc2 * zc1 - zc2 * xc1 ;
M(12,12) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(12,13) =-xc2 * zc1 + zc2 * xc1 ;
M(12,29) =-xc2 * yc1 ;
M(12,30) =-2 * zc2 * yc1 ;
M(12,31) =xc2 * yc1 - yc2 * xc1 ;
M(13,1) =zc2 * yc1 ;
M(13,2) =-2 * xc2 * yc1 ;
M(13,3) =yc2 * zc1 - zc2 * yc1 ;
M(13,10) =xc2 * zc1 - zc2 * xc1 ;
M(13,11) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(13,12) =-xc2 * zc1 + zc2 * xc1 ;
M(13,28) =-xc2 * yc1 ;
M(13,29) =-2 * zc2 * yc1 ;
M(13,30) =xc2 * yc1 - yc2 * xc1 ;
M(14,0) =zc2 * yc1 ;
M(14,1) =-2 * xc2 * yc1 ;
M(14,2) =yc2 * zc1 - zc2 * yc1 ;
M(14,9) =xc2 * zc1 - zc2 * xc1 ;
M(14,10) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(14,11) =-xc2 * zc1 + zc2 * xc1 ;
M(14,27) =-xc2 * yc1 ;
M(14,28) =-2 * zc2 * yc1 ;
M(14,29) =xc2 * yc1 - yc2 * xc1 ;
M(15,59) =zb2 * yb1 ;
M(15,60) =-2 * xb2 * yb1 ;
M(15,61) =yb2 * zb1 - zb2 * yb1 ;
M(15,67) =xb2 * zb1 - zb2 * xb1 ;
M(15,68) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(15,69) =-xb2 * zb1 + zb2 * xb1 ;
M(15,75) =-xb2 * yb1 ;
M(9,7) = yc2 * zc1 - zc2 * yc1 ;
M(9,14) =xc2 * zc1 - zc2 * xc1 ;
M(9,15) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(9,16) =-xc2 * zc1 + zc2 * xc1 ;
M(9,32) =-xc2 * yc1 ;
M(9,33) =-2 * zc2 * yc1 ;
M(9,34) =xc2 * yc1 - yc2 * xc1 ;
M(10,4) =zc2 * yc1 ;
M(10,5) =-2 * xc2 * yc1 ;
M(10,6) =yc2 * zc1 - zc2 * yc1 ;
M(10,13) =xc2 * zc1 - zc2 * xc1 ;
M(10,14) =2 * xc2 * xc1 + 2 * zc2 * zc1 ;
M(10,15) =-xc2 * zc1 + zc2 * xc1 ;
M(10,31) =-xc2 * yc1 ;
M(10,32) =-2 * zc2 * yc1 ;
M(10,33) =xc2 * yc1 - yc2 * xc1 ;
M(11,3) =zc2 * yc1 ;
M(16,74) =-xb2 * yb1 ;
M(16,75) =-2 * zb2 * yb1 ;
M(16,76) =xb2 * yb1 - yb2 * xb1 ;
M(17,57) =zb2 * yb1 ;
M(17,58) =-2 * xb2 * yb1 ;
M(17,59) =yb2 * zb1 - zb2 * yb1 ;
M(17,65) =xb2 * zb1 - zb2 * xb1 ;
M(17,66) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(17,67) =-xb2 * zb1 + zb2 * xb1 ;
M(17,73) =-xb2 * yb1 ;
M(17,74) =-2 * zb2 * yb1 ;
M(17,75) =xb2 * yb1 - yb2 * xb1 ;
M(18,56) =zb2 * yb1 ;
M(18,57) =-2 * xb2 * yb1 ;
M(18,58) =yb2 * zb1 - zb2 * yb1 ;
M(18,64) =xb2 * zb1 - zb2 * xb1 ;
M(18,65) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(18,66) =-xb2 * zb1 + zb2 * xb1 ;
M(18,72) =-xb2 * yb1 ;
M(18,73) =-2 * zb2 * yb1 ;
M(18,74) =xb2 * yb1 - yb2 * xb1 ;
M(19,55) =zb2 * yb1 ;
M(19,56) =-2 * xb2 * yb1 ;
M(19,57) =yb2 * zb1 - zb2 * yb1 ;
M(19,63) =xb2 * zb1 - zb2 * xb1 ;
M(19,64) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(19,65) =-xb2 * zb1 + zb2 * xb1 ;
M(19,71) =-xb2 * yb1 ;
M(19,72) =-2 * zb2 * yb1 ;
M(19,73) =xb2 * yb1 - yb2 * xb1 ;
M(20,54) =zb2 * yb1 ;
M(20,55) =-2 * xb2 * yb1 ;
M(20,56) =yb2 * zb1 - zb2 * yb1 ;
M(20,62) =xb2 * zb1 - zb2 * xb1 ;
M(20,63) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(20,64) =-xb2 * zb1 + zb2 * xb1 ;
M(20,70) =-xb2 * yb1 ;
M(20,71) =-2 * zb2 * yb1 ;
M(20,72) =xb2 * yb1 - yb2 * xb1 ;
M(21,28) =zb2 * yb1 ;
M(21,29) =-2 * xb2 * yb1 ;
M(21,30) =yb2 * zb1 - zb2 * yb1 ;
```

```
M(15,76) =-2 * zb2 * yb1 ;
M(15,77) =xb2 * yb1 - yb2 * xb1 ;
M(16,58) =zb2 * yb1 ;
M(16,59) =-2 * xb2 * yb1 ;
M(16,60) =yb2 * zb1 - zb2 * yb1 ;
M(16,66) =xb2 * zb1 - zb2 * xb1 ;
M(16,67) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(16,68) =-xb2 * zb1 + zb2 * xb1 ;
M(22,29) =yb2 * zb1 - zb2 * yb1 ;
M(22,36) =xb2 * zb1 - zb2 * xb1 ;
M(22,37) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(22,38) =-xb2 * zb1 + zb2 * xb1 ;
M(22,45) =-xb2 * yb1 ;
M(22,46) =-2 * zb2 * yb1 ;
M(22,47) =xb2 * yb1 - yb2 * xb1 ;
M(23,15) =zb2 * yb1 ;
M(23,16) =-2 * xb2 * yb1 ;
M(23,17) =yb2 * zb1 - zb2 * yb1 ;
M(23,24) =xb2 * zb1 - zb2 * xb1 ;
M(23,25) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(23,26) =-xb2 * zb1 + zb2 * xb1 ;
M(23,42) =-xb2 * yb1 ;
M(23,43) =-2 * zb2 * yb1 ;
M(23,44) =xb2 * yb1 - yb2 * xb1 ;
M(24,14) =zb2 * yb1 ;
M(24,15) =-2 * xb2 * yb1 ;
M(24,16) =yb2 * zb1 - zb2 * yb1 ;
M(24,23) =xb2 * zb1 - zb2 * xb1 ;
M(24,24) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(24,25) =-xb2 * zb1 + zb2 * xb1 ;
M(24,41) =-xb2 * yb1 ;
M(24,42) =-2 * zb2 * yb1 ;
M(24,43) =xb2 * yb1 - yb2 * xb1 ;
M(25,13) =zb2 * yb1 ;
M(25,14) =-2 * xb2 * yb1 ;
M(25,15) =yb2 * zb1 - zb2 * yb1 ;
M(25,22) =xb2 * zb1 - zb2 * xb1 ;
M(25,23) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(25,24) =-xb2 * zb1 + zb2 * xb1 ;
M(25,40) =-xb2 * yb1 ;
M(25,41) =-2 * zb2 * yb1 ;
M(25,42) =xb2 * yb1 - yb2 * xb1 ;
M(26,12) =zb2 * yb1 ;
M(26,13) =-2 * xb2 * yb1 ;
M(26,14) =yb2 * zb1 - zb2 * yb1 ;
M(26,21) =xb2 * zb1 - zb2 * xb1 ;
M(26,22) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(26,23) =-xb2 * zb1 + zb2 * xb1 ;
M(26,39) =-xb2 * yb1 ;
M(26,40) =-2 * zb2 * yb1 ;
M(26,41) =xb2 * yb1 - yb2 * xb1 ;
M(27,11) =zb2 * yb1 ;
M(27,12) =-2 * xb2 * yb1 ;
M(27,13) =yb2 * zb1 - zb2 * yb1 ;
M(27,20) =xb2 * zb1 - zb2 * xb1 ;
M(27,21) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(27,22) =-xb2 * zb1 + zb2 * xb1 ;
M(27,38) =-xb2 * yb1 ;
M(33,12) =xb2 * zb1 - zb2 * xb1 ;
M(21,37) =xb2 * zb1 - zb2 * xb1 ;
M(21,38) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(21,39) =-xb2 * zb1 + zb2 * xb1 ;
M(21,46) =-xb2 * yb1 ;
M(21,47) =-2 * zb2 * yb1 ;
M(21,48) =xb2 * yb1 - yb2 * xb1 ;
M(22,27) =zb2 * yb1 ;
M(22,28) =-2 * xb2 * yb1 ;
M(27,39) =-2 * zb2 * yb1 ;
M(27,40) =xb2 * yb1 - yb2 * xb1 ;
M(28,10) =zb2 * yb1 ;
M(28,11) =-2 * xb2 * yb1 ;
M(28,12) =yb2 * zb1 - zb2 * yb1 ;
M(28,19) =xb2 * zb1 - zb2 * xb1 ;
M(28,20) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(28,21) =-xb2 * zb1 + zb2 * xb1 ;
M(28,37) =-xb2 * yb1 ;
M(28,38) =-2 * zb2 * yb1 ;
M(28,39) =xb2 * yb1 - yb2 * xb1 ;
M(29,9) =zb2 * yb1 ;
M(29,10) =-2 * xb2 * yb1 ;
M(29,11) =yb2 * zb1 - zb2 * yb1 ;
M(29,18) =xb2 * zb1 - zb2 * xb1 ;
M(29,19) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(29,20) =-xb2 * zb1 + zb2 * xb1 ;
M(29,36) =-xb2 * yb1 ;
M(29,37) =-2 * zb2 * yb1 ;
M(29,38) =xb2 * yb1 - yb2 * xb1 ;
M(30,6) =zb2 * yb1 ;
M(30,7) =-2 * xb2 * yb1 ;
M(30,8) =yb2 * zb1 - zb2 * yb1 ;
M(30,15) =xb2 * zb1 - zb2 * xb1 ;
M(30,16) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(30,17) =-xb2 * zb1 + zb2 * xb1 ;
M(30,33) =-xb2 * yb1 ;
M(30,34) =-2 * zb2 * yb1 ;
M(30,35) =xb2 * yb1 - yb2 * xb1 ;
M(31,5) =zb2 * yb1 ;
M(31,6) =-2 * xb2 * yb1 ;
M(31,7) =yb2 * zb1 - zb2 * yb1 ;
M(31,14) =xb2 * zb1 - zb2 * xb1 ;
M(31,15) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(31,16) =-xb2 * zb1 + zb2 * xb1 ;
M(31,32) =-xb2 * yb1 ;
M(31,33) =-2 * zb2 * yb1 ;
M(31,34) =xb2 * yb1 - yb2 * xb1 ;
M(32,4) =zb2 * yb1 ;
M(32,5) =-2 * xb2 * yb1 ;
M(32,6) =yb2 * zb1 - zb2 * yb1 ;
M(32,13) =xb2 * zb1 - zb2 * xb1 ;
M(32,14) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(32,15) =-xb2 * zb1 + zb2 * xb1 ;
M(32,31) =-xb2 * yb1 ;
M(32,32) =-2 * zb2 * yb1 ;
M(32,33) =xb2 * yb1 - yb2 * xb1 ;
M(33,3) =zb2 * yb1 ;
M(33,4) =-2 * xb2 * yb1 ;
M(33,5) =yb2 * zb1 - zb2 * yb1 ;
M(38,76) =xa2 * ya1 - ya2 * xa1 ;
```

```
M(33,13) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(33,14) =-xb2 * zb1 + zb2 * xb1 ;
M(33,30) =-xb2 * yb1 ;
M(33,31) =-2 * zb2 * yb1 ;
M(33,32) =xb2 * yb1 - yb2 * xb1 ;
M(34,2) =zb2 * yb1 ;
M(34,3) =-2 * xb2 * yb1 ;
M(34,4) =yb2 * zb1 - zb2 * yb1 ;
M(34,11) =xb2 * zb1 - zb2 * xb1 ;
M(34,12) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(34,13) =-xb2 * zb1 + zb2 * xb1 ;
M(34,29) =-xb2 * yb1 ;
M(34,30) =-2 * zb2 * yb1 ;
M(34,31) =xb2 * yb1 - yb2 * xb1 ;
M(35,1) =zb2 * yb1 ;
M(35,2) =-2 * xb2 * yb1 ;
M(35,3) =yb2 * zb1 - zb2 * yb1 ;
M(35,10) =xb2 * zb1 - zb2 * xb1 ;
M(35,11) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(35,12) =-xb2 * zb1 + zb2 * xb1 ;
M(35,28) =-xb2 * yb1 ;
M(35,29) =-2 * zb2 * yb1 ;
M(35,30) =xb2 * yb1 - yb2 * xb1 ;
M(36,0) =zb2 * yb1 ;
M(36,1) =-2 * xb2 * yb1 ;
M(36,2) =yb2 * zb1 - zb2 * yb1 ;
M(36,9) =xb2 * zb1 - zb2 * xb1 ;
M(36,10) =2 * xb2 * xb1 + 2 * zb2 * zb1 ;
M(36,11) =-xb2 * zb1 + zb2 * xb1 ;
M(36,27) =-xb2 * yb1 ;
M(36,28) =-2 * zb2 * yb1 ;
M(36,29) =xb2 * yb1 - yb2 * xb1 ;
M(37,59) =za2 * ya1 ;
M(37,60) =-2 * xa2 * ya1 ;
M(37,61) =ya2 * za1 - za2 * ya1 ;
M(37,67) =xa2 * za1 - za2 * xa1 ;
M(37,68) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(37,69) =-xa2 * za1 + za2 * xa1 ;
M(37,75) =-xa2 * ya1 ;
M(37,76) =-2 * za2 * ya1 ;
M(37,77) =xa2 * ya1 - ya2 * xa1 ;
M(38,58) =za2 * ya1 ;
M(38,59) =-2 * xa2 * ya1 ;
M(38,60) =ya2 * za1 - za2 * ya1 ;
M(38,66) =xa2 * za1 - za2 * xa1 ;
M(38,67) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(38,68) =-xa2 * za1 + za2 * xa1 ;
M(38,74) =-xa2 * ya1 ;
M(38,75) =-2 * za2 * ya1 ;
M(44,42) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(44,43) =-xa2 * za1 + za2 * xa1 ;
M(44,50) =-xa2 * ya1 ;
M(44,51) =-2 * za2 * ya1 ;
M(44,52) =xa2 * ya1 - ya2 * xa1 ;
M(45,31) =za2 * ya1 ;
M(45,32) =-2 * xa2 * ya1 ;
M(45,33) =ya2 * za1 - za2 * ya1 ;
M(45,40) =xa2 * za1 - za2 * xa1 ;
M(45,41) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(39,57) =za2 * ya1 ;
M(39,58) =-2 * xa2 * ya1 ;
M(39,59) =ya2 * za1 - za2 * ya1 ;
M(39,65) =xa2 * za1 - za2 * xa1 ;
M(39,66) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(39,67) =-xa2 * za1 + za2 * xa1 ;
M(39,73) =-xa2 * ya1 ;
M(39,74) =-2 * za2 * ya1 ;
M(39,75) =xa2 * ya1 - ya2 * xa1 ;
M(40,56) =za2 * ya1 ;
M(40,57) =-2 * xa2 * ya1 ;
M(40,58) =ya2 * za1 - za2 * ya1 ;
M(40,64) =xa2 * za1 - za2 * xa1 ;
M(40,65) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(40,66) =-xa2 * za1 + za2 * xa1 ;
M(40,72) =-xa2 * ya1 ;
M(40,73) =-2 * za2 * ya1 ;
M(40,74) =xa2 * ya1 - ya2 * xa1 ;
M(41,55) =za2 * ya1 ;
M(41,56) =-2 * xa2 * ya1 ;
M(41,57) =ya2 * za1 - za2 * ya1 ;
M(41,63) =xa2 * za1 - za2 * xa1 ;
M(41,64) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(41,65) =-xa2 * za1 + za2 * xa1 ;
M(41,71) =-xa2 * ya1 ;
M(41,72) =-2 * za2 * ya1 ;
M(41,73) =xa2 * ya1 - ya2 * xa1 ;
M(42,54) =za2 * ya1 ;
M(42,55) =-2 * xa2 * ya1 ;
M(42,56) =ya2 * za1 - za2 * ya1 ;
M(42,62) =xa2 * za1 - za2 * xa1 ;
M(42,63) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(42,64) =-xa2 * za1 + za2 * xa1 ;
M(42,70) =-xa2 * ya1 ;
M(42,71) =-2 * za2 * ya1 ;
M(42,72) =xa2 * ya1 - ya2 * xa1 ;
M(43,33) =za2 * ya1 ;
M(43,34) =-2 * xa2 * ya1 ;
M(43,35) =ya2 * za1 - za2 * ya1 ;
M(43,42) =xa2 * za1 - za2 * xa1 ;
M(43,43) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(43,44) =-xa2 * za1 + za2 * xa1 ;
M(43,51) =-xa2 * ya1 ;
M(43,52) =-2 * za2 * ya1 ;
M(43,53) =xa2 * ya1 - ya2 * xa1 ;
M(44,32) =za2 * ya1 ;
M(44,33) =-2 * xa2 * ya1 ;
M(44,34) =ya2 * za1 - za2 * ya1 ;
M(44,41) =xa2 * za1 - za2 * xa1 ;
M(50,15) =za2 * ya1 ;
M(50,16) =-2 * xa2 * ya1 ;
M(50,17) =ya2 * za1 - za2 * ya1 ;
M(50,24) =xa2 * za1 - za2 * xa1 ;
M(50,25) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(50,26) =-xa2 * za1 + za2 * xa1 ;
M(50,42) =-xa2 * ya1 ;
M(50,43) =-2 * za2 * ya1 ;
M(50,44) =xa2 * ya1 - ya2 * xa1 ;
M(51,14) =za2 * ya1 ;
```

```
M(45,42) =-xa2 * za1 + za2 * xa1;
M(45,49) =-xa2 * ya1;
M(45,50) =-2 * za2 * ya1;
M(45,51) =xa2 * ya1 - ya2 * xa1;
M(46,30) =za2 * ya1;
M(46,31) =-2 * xa2 * ya1;
M(46,32) =ya2 * za1 - za2 * ya1;
M(46,39) =xa2 * za1 - za2 * xa1;
M(46,40) =2 * xa2 * xa1 + 2 * za2 * za1;
M(46,41) =-xa2 * za1 + za2 * xa1;
M(46,48) =-xa2 * ya1;
M(46,49) =-2 * za2 * ya1;
M(46,50) =xa2 * ya1 - ya2 * xa1;
M(47,29) =za2 * ya1;
M(47,30) =-2 * xa2 * ya1;
M(47,31) =ya2 * za1 - za2 * ya1;
M(47,38) =xa2 * za1 - za2 * xa1;
M(47,39) =2 * xa2 * xa1 + 2 * za2 * za1;
M(47,40) =-xa2 * za1 + za2 * xa1;
M(47,47) =-xa2 * ya1;
M(47,48) =-2 * za2 * ya1;
M(47,49) =xa2 * ya1 - ya2 * xa1;
M(48,28) =za2 * ya1;
M(48,29) =-2 * xa2 * ya1;
M(48,30) =ya2 * za1 - za2 * ya1;
M(48,37) =xa2 * za1 - za2 * xa1;
M(48,38) =2 * xa2 * xa1 + 2 * za2 * za1;
M(48,39) =-xa2 * za1 + za2 * xa1;
M(48,46) =-xa2 * ya1;
M(48,47) =-2 * za2 * ya1;
M(48,48) =xa2 * ya1 - ya2 * xa1;
M(49,27) =za2 * ya1;
M(49,28) =-2 * xa2 * ya1;
M(49,29) =ya2 * za1 - za2 * ya1;
M(49,36) =xa2 * za1 - za2 * xa1;
M(49,37) =2 * xa2 * xa1 + 2 * za2 * za1;
M(49,38) =-xa2 * za1 + za2 * xa1;
M(49,45) =-xa2 * ya1;
M(49,46) =-2 * za2 * ya1;
M(49,47) =xa2 * ya1 - ya2 * xa1;
M(55,21) =-xa2 * za1 + za2 * xa1;
M(55,37) =-xa2 * ya1;
M(55,38) =-2 * za2 * ya1;
M(55,39) =xa2 * ya1 - ya2 * xa1;
M(56,9) =za2 * ya1;
M(56,10) =-2 * xa2 * ya1;
M(56,11) =ya2 * za1 - za2 * ya1;
M(56,18) =xa2 * za1 - za2 * xa1;
M(56,19) =2 * xa2 * xa1 + 2 * za2 * za1;
M(56,20) =-xa2 * za1 + za2 * xa1;
M(56,36) =-xa2 * ya1;
M(56,37) =-2 * za2 * ya1;
M(56,38) =xa2 * ya1 - ya2 * xa1;
M(57,6) =za2 * ya1;
M(57,7) =-2 * xa2 * ya1;
M(57,8) =ya2 * za1 - za2 * ya1;
M(57,15) =xa2 * za1 - za2 * xa1;
M(57,16) =2 * xa2 * xa1 + 2 * za2 * za1;
M(57,17) =-xa2 * za1 + za2 * xa1;
M(51,15) =-2 * xa2 * ya1;
M(51,16) =ya2 * za1 - za2 * ya1;
M(51,23) =xa2 * za1 - za2 * xa1;
M(51,24) =2 * xa2 * xa1 + 2 * za2 * za1;
M(51,25) =-xa2 * za1 + za2 * xa1;
M(51,41) =-xa2 * ya1;
M(51,42) =-2 * za2 * ya1;
M(51,43) =xa2 * ya1 - ya2 * xa1;
M(52,13) =za2 * ya1;
M(52,14) =-2 * xa2 * ya1;
M(52,15) =ya2 * za1 - za2 * ya1;
M(52,22) =xa2 * za1 - za2 * xa1;
M(52,23) =2 * xa2 * xa1 + 2 * za2 * za1;
M(52,24) =-xa2 * za1 + za2 * xa1;
M(52,40) =-xa2 * ya1;
M(52,41) =-2 * za2 * ya1;
M(52,42) =xa2 * ya1 - ya2 * xa1;
M(53,12) =za2 * ya1;
M(53,13) =-2 * xa2 * ya1;
M(53,14) =ya2 * za1 - za2 * ya1;
M(53,21) =xa2 * za1 - za2 * xa1;
M(53,22) =2 * xa2 * xa1 + 2 * za2 * za1;
M(53,23) =-xa2 * za1 + za2 * xa1;
M(53,39) =-xa2 * ya1;
M(53,40) =-2 * za2 * ya1;
M(53,41) =xa2 * ya1 - ya2 * xa1;
M(54,11) =za2 * ya1;
M(54,12) =-2 * xa2 * ya1;
M(54,13) =ya2 * za1 - za2 * ya1;
M(54,20) =xa2 * za1 - za2 * xa1;
M(54,21) =2 * xa2 * xa1 + 2 * za2 * za1;
M(54,22) =-xa2 * za1 + za2 * xa1;
M(54,38) =-xa2 * ya1;
M(54,39) =-2 * za2 * ya1;
M(54,40) =xa2 * ya1 - ya2 * xa1;
M(55,10) =za2 * ya1;
M(55,11) =-2 * xa2 * ya1;
M(55,12) =ya2 * za1 - za2 * ya1;
M(55,19) =xa2 * za1 - za2 * xa1;
M(55,20) =2 * xa2 * xa1 + 2 * za2 * za1;
M(61,3) =-2 * xa2 * ya1;
M(61,4) =ya2 * za1 - za2 * ya1;
M(61,11) =xa2 * za1 - za2 * xa1;
M(61,12) =2 * xa2 * xa1 + 2 * za2 * za1;
M(61,13) =-xa2 * za1 + za2 * xa1;
M(61,29) =-xa2 * ya1;
M(61,30) =-2 * za2 * ya1;
M(61,31) =xa2 * ya1 - ya2 * xa1;
M(62,1) =za2 * ya1;
M(62,2) =-2 * xa2 * ya1;
M(62,3) =ya2 * za1 - za2 * ya1;
M(62,10) =xa2 * za1 - za2 * xa1;
M(62,11) =2 * xa2 * xa1 + 2 * za2 * za1;
M(62,12) =-xa2 * za1 + za2 * xa1;
M(62,28) =-xa2 * ya1;
M(62,29) =-2 * za2 * ya1;
M(62,30) =xa2 * ya1 - ya2 * xa1;
M(63,0) =za2 * ya1;
M(63,1) =-2 * xa2 * ya1;
```

```
M(57,33) =-xa2 * ya1 ;
M(57,34) =-2 * za2 * ya1 ;
M(57,35) =xa2 * ya1 - ya2 * xa1 ;
M(58,5) =za2 * ya1 ;
M(58,6) =-2 * xa2 * ya1 ;
M(58,7) =ya2 * za1 - za2 * ya1 ;
M(58,14) =xa2 * za1 - za2 * xa1 ;
M(58,15) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(58,16) =-xa2 * za1 + za2 * xa1 ;
M(58,32) =-xa2 * ya1 ;
M(58,33) =-2 * za2 * ya1 ;
M(58,34) =xa2 * ya1 - ya2 * xa1 ;
M(59,4) =za2 * ya1 ;
M(59,5) =-2 * xa2 * ya1 ;
M(59,6) =ya2 * za1 - za2 * ya1 ;
M(59,13) =xa2 * za1 - za2 * xa1 ;
M(59,14) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(59,15) =-xa2 * za1 + za2 * xa1 ;
M(59,31) =-xa2 * ya1 ;
M(59,32) =-2 * za2 * ya1 ;
M(59,33) =xa2 * ya1 - ya2 * xa1 ;
M(60,3) =za2 * ya1 ;
M(60,4) =-2 * xa2 * ya1 ;
M(60,5) =ya2 * za1 - za2 * ya1 ;
M(60,12) =xa2 * za1 - za2 * xa1 ;
M(60,13) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(60,14) =-xa2 * za1 + za2 * xa1 ;
M(60,30) =-xa2 * ya1 ;
M(60,31) =-2 * za2 * ya1 ;
M(60,32) =xa2 * ya1 - ya2 * xa1 ;
M(61,2) =za2 * ya1 ;
M(70,2) =1 ;
M(70,20) =1 ;
M(70,47) =1 ;
M(70,78) =-1 ;
```

```
M(63,2) =ya2 * za1 - za2 * ya1 ;
M(63,9) =xa2 * za1 - za2 * xa1 ;
M(63,10) =2 * xa2 * xa1 + 2 * za2 * za1 ;
M(63,11) =-xa2 * za1 + za2 * xa1 ;
M(63,27) =-xa2 * ya1 ;
M(63,28) =-2 * za2 * ya1 ;
M(63,29) =xa2 * ya1 - ya2 * xa1 ;
M(64,8) =1 ;
M(64,26) =1 ;
M(64,53) =1 ;
M(64,84) =-1 ;
M(65,7) =1 ;
M(65,25) =1 ;
M(65,52) =1 ;
M(65,83) =-1 ;
M(66,6) =1 ;
M(66,24) =1 ;
M(66,51) =1 ;
M(66,82) =-1 ;
M(67,5) =1 ;
M(67,23) =1 ;
M(67,50) =1 ;
M(67,81) =-1 ;
M(68,4) =1 ;
M(68,22) =1 ;
M(68,49) =1 ;
M(68,80) =-1 ;
M(69,3) =1 ;
M(69,21) =1 ;
M(69,48) =1 ;
M(69,79) =-1 ;
```

```
M= EliminationGauss(M) ;
```

```
h1=M(0,79) ;  k9=M(8,81) ; m17=M(16,83) ; h26=M(25,79) ; k34=M(33,81) ; m42=M(41,83) ; h51=M(50,79) ;
i1=M(0,80) ;  l9=M(8,82) ; n17=M(16,84) ; i26=M(25,80) ; l34=M(33,82) ; n42=M(41,84) ; i51=M(50,80) ;
k1=M(0,81) ;  m9=M(8,83) ; h18=M(17,79) ; k26=M(25,81) ; m34=M(33,83) ; h43=M(42,79) ; k51=M(50,81) ;
l1=M(0,82) ;  n9=M(8,84) ; i18=M(17,80) ; l26=M(25,82) ; n34=M(33,84) ; i43=M(42,80) ; l51=M(50,82) ;
m1=M(0,83) ; h10=M(9,79) ; k18=M(17,81) ; m26=M(25,83) ; h35=M(34,79) ; k43=M(42,81) ; m51=M(50,83) ;
n1=M(0,84) ; i10=M(9,80) ; l18=M(17,82) ; n26=M(25,84) ; i35=M(34,80) ; l43=M(42,82) ; n51=M(50,84) ;
h2=M(1,79) ; k10=M(9,81) ; m18=M(17,83) ; h27=M(26,79) ; k35=M(34,81) ; m43=M(42,83) ; h52=M(51,79) ;
i2=M(1,80) ; l10=M(9,82) ; n18=M(17,84) ; i27=M(26,80) ; l35=M(34,82) ; n43=M(42,84) ; i52=M(51,80) ;
k2=M(1,81) ; m10=M(9,83) ; h19=M(18,79) ; k27=M(26,81) ; m35=M(34,83) ; h44=M(43,79) ; k52=M(51,81) ;
l2=M(1,82) ; n10=M(9,84) ; i19=M(18,80) ; l27=M(26,82) ; n35=M(34,84) ; i44=M(43,80) ; l52=M(51,82) ;
m2=M(1,83) ; h11=M(10,79) ; k19=M(18,81) ; m27=M(26,83) ; h36=M(35,79) ; k44=M(43,81) ; m52=M(51,83) ;
n2=M(1,84) ; i11=M(10,80) ; l19=M(18,82) ; n27=M(26,84) ; i36=M(35,80) ; l44=M(43,82) ; n52=M(51,84) ;
h3=M(2,79) ; k11=M(10,81) ; m19=M(18,83) ; h28=M(27,79) ; k36=M(35,81) ; m44=M(43,83) ; h53=M(52,79) ;
i3=M(2,80) ; l11=M(10,82) ; n19=M(18,84) ; i28=M(27,80) ; l36=M(35,82) ; n44=M(43,84) ; i53=M(52,80) ;
k3=M(2,81) ; m11=M(10,83) ; h20=M(19,79) ; k28=M(27,81) ; m36=M(35,83) ; h45=M(44,79) ; k53=M(52,81) ;
l3=M(2,82) ; n11=M(10,84) ; i20=M(19,80) ; l28=M(27,82) ; n36=M(35,84) ; i45=M(44,80) ; l53=M(52,82) ;
m3=M(2,83) ; h12=M(11,79) ; k20=M(19,81) ; m28=M(27,83) ; h37=M(36,79) ; k45=M(44,81) ; m53=M(52,83) ;
n3=M(2,84) ; i12=M(11,80) ; l20=M(19,82) ; n28=M(27,84) ; i37=M(36,80) ; l45=M(44,82) ; n53=M(52,84) ;
h4=M(3,79) ; k12=M(11,81) ; m20=M(19,83) ; h29=M(28,79) ; k37=M(36,81) ; m45=M(44,83) ; h54=M(53,79) ;
```

i4=M(3,80); l12=M(11,82); n20=M(19,84); i29=M(28,80); l37=M(36,82); n45=M(44,84); i54=M(53,80);
k4=M(3,81); m12=M(11,83); h21=M(20,79); k29=M(28,81); m37=M(36,83); h46=M(45,79); k54=M(53,81);
l4=M(3,82); n12=M(11,84); i21=M(20,80); l29=M(28,82); n37=M(36,84); i46=M(45,80); l54=M(53,82);
m4=M(3,83); h13=M(12,79); k21=M(20,81); m29=M(28,83); h38=M(37,79); k46=M(45,81); m54=M(53,83);
n4=M(3,84); i13=M(12,80); l21=M(20,82); n29=M(28,84); i38=M(37,80); l46=M(45,82); n54=M(53,84);
h5=M(4,79); k13=M(12,81); m21=M(20,83); h30=M(29,79); k38=M(37,81); m46=M(45,83); a1=M(54,70);
i5=M(4,80); l13=M(12,82); n21=M(20,84); i30=M(29,80); l38=M(37,82); n46=M(45,84); b1=M(54,71);
k5=M(4,81); m13=M(12,83); h22=M(21,79); k30=M(29,81); m38=M(37,83); h47=M(46,79); c1=M(54,72);
l5=M(4,82); n13=M(12,84); i22=M(21,80); l30=M(29,82); n38=M(37,84); i47=M(46,80); d1=M(54,73);
m5=M(4,83); h14=M(13,79); k22=M(21,81); m30=M(29,83); h39=M(38,79); k47=M(46,81); e1=M(54,74);
n5=M(4,84); i14=M(13,80); l22=M(21,82); n30=M(29,84); i39=M(38,80); l47=M(46,82); f1=M(54,75);
h6=M(5,79); k14=M(13,81); m22=M(21,83); h31=M(30,79); k39=M(38,81); m47=M(46,83); a2=M(55,70);
i6=M(5,80); l14=M(13,82); n22=M(21,84); i31=M(30,80); l39=M(38,82); n47=M(46,84); b2=M(55,71);
k6=M(5,81); m14=M(13,83); h23=M(22,79); k31=M(30,81); m39=M(38,83); h48=M(47,79); c2=M(55,72);
l6=M(5,82); n14=M(13,84); i23=M(22,80); l31=M(30,82); n39=M(38,84); i48=M(47,80); d2=M(55,73);
m6=M(5,83); h15=M(14,79); k23=M(22,81); m31=M(30,83); h40=M(39,79); k48=M(47,81); e2=M(55,74);
n6=M(5,84); i15=M(14,80); l23=M(22,82); n31=M(30,84); i40=M(39,80); l48=M(47,82); f2=M(55,75);
h7=M(6,79); k15=M(14,81); m23=M(22,83); h32=M(31,79); k40=M(39,81); m48=M(47,83); a3=M(56,70);
i7=M(6,80); l15=M(14,82); n23=M(22,84); i32=M(31,80); l40=M(39,82); n48=M(47,84); b3=M(56,71);
k7=M(6,81); m15=M(14,83); h24=M(23,79); k32=M(31,81); m40=M(39,83); h49=M(48,79); c3=M(56,72);
l7=M(6,82); n15=M(14,84); i24=M(23,80); l32=M(31,82); n40=M(39,84); i49=M(48,80); d3=M(56,73);
m7=M(6,83); h16=M(15,79); k24=M(23,81); m32=M(31,83); h41=M(40,79); k49=M(48,81); e3=M(56,74);
n7=M(6,84); i16=M(15,80); l24=M(23,82); n32=M(31,84); i41=M(40,80); l49=M(48,82); f3=M(56,75);
h8=M(7,79); k16=M(15,81); m24=M(23,83); h33=M(32,79); k41=M(40,81); m49=M(48,83); a4=M(57,70);
i8=M(7,80); l16=M(15,82); n24=M(23,84); i33=M(32,80); l41=M(40,82); n49=M(48,84); b4=M(57,71);
k8=M(7,81); m16=M(15,83); h25=M(24,79); k33=M(32,81); m41=M(40,83); h50=M(49,79); c4=M(57,72);
l8=M(7,82); n16=M(15,84); i25=M(24,80); l33=M(32,82); n41=M(40,84); i50=M(49,80); d4=M(57,73);
m8=M(7,83); h17=M(16,79); k25=M(24,81); m33=M(32,83); h42=M(41,79); k50=M(49,81); e4=M(57,74);
n8=M(7,84); i17=M(16,80); l25=M(24,82); n33=M(32,84); i42=M(41,80); l50=M(49,82); f4=M(57,75);
h9=M(8,79); k17=M(16,81); m25=M(24,83); h34=M(33,79); k42=M(41,81); m50=M(49,83); a5=M(58,70);
i9=M(8,80); l17=M(16,82); n25=M(24,84); i34=M(33,80); l42=M(41,82); n50=M(49,84); b5=M(58,71);
c5=M(58,72); b13=M(66,71); b7=M(60,71); a15=M(68,70); f8=M(61,75); e16=M(69,74); d10=M(63,73);
d5=M(58,73); c13=M(66,72); c7=M(60,72); b15=M(68,71); e2=M(61,76); f16=M(69,75); e10=M(63,74);
e5=M(58,74); d13=M(66,73); d7=M(60,73); c15=M(68,72); g1=M(61,77); e4=M(69,76); f10=M(63,75);
f5=M(58,75); e13=M(66,74); e7=M(60,74); d15=M(68,73); a9=M(62,70); g2=M(69,77); a11=M(64,70);
a6=M(59,70); f13=M(66,75); f7=M(60,75); e15=M(68,74); b9=M(62,71); h55=M(70,79); b11=M(64,71);
b6=M(59,71); a14=M(67,70); e1=M(60,76); f15=M(68,75); c9=M(62,72); i55=M(70,80); c11=M(64,72);
c6=M(59,72); b14=M(67,71); a8=M(61,70); e3=M(68,76); d9=M(62,73); k55=M(70,81); d11=M(64,73);
d6=M(59,73); c14=M(67,72); b8=M(61,71); a16=M(69,70); e9=M(62,74); l55=M(70,82); e11=M(64,74);
e6=M(59,74); d14=M(67,73); c8=M(61,72); b16=M(69,71); f9=M(62,75); m55=M(70,83); f11=M(64,75);
f6=M(59,75); e14=M(67,74); d8=M(61,73); c16=M(69,72); a10=M(63,70); n55=M(70,84); a12=M(65,70);
a7=M(60,70); f14=M(67,75); e8=M(61,74); d16=M(69,73); b10=M(63,71); c10=M(63,72); b12=M(65,71);
e10=M(63,74); e10=M(63,74); f10=M(63,75); f10=M(63,75); c12=M(65,72); a12=M(65,70); f11=M(64,75);
f10=M(63,75); f10=M(63,75); a11=M(64,70); a11=M(64,70); d12=M(65,73); b12=M(65,71); e11=M(64,74);
a11=M(64,70); a11=M(64,70); c11=M(64,72); c11=M(64,72); e12=M(65,74); d11=M(64,73); a13=M(66,70);
b11=M(64,71); c11=M(64,72); d11=M(64,73); e11=M(64,74); f12=M(65,75); f11=M(64,75); e11=M(64,74);
c11=M(64,72); d11=M(64,73);

Matrice ANGLE(12,12)
ANGLE=0
// ANGLE

ANGLE(0,1) = 1; ANGLE(5,5) = -h55;
ANGLE(1,2) = 1; ANGLE(6,7) = 1;
ANGLE(2,3) = 1; ANGLE(7,8) = 1;
ANGLE(3,4) = 1; ANGLE(8,9) = 1;
ANGLE(4,5) = 1; ANGLE(9,10) = 1;
ANGLE(5,0) = -n55; ANGLE(10,11) = 1;

ANGLE(5,1) = -m55 ; ANGLE(11,6) = -n55 ;
ANGLE(5,2) = -l55 ; ANGLE(11,7) = -m55 ;
ANGLE(5,3) = -k55 ; ANGLE(11,8) = -l55 ;
ANGLE(5,4) = -i55 ; ANGLE(11,9) = -k55 ;
ANGLE(11,11) = -h55 ; ANGLE(11,10) = -i55 ;

calculValeursPropres(ANGLE,sol-Angle)

Matrice TRANSX(12,12)
TRANSX=0

// translation

TRANSX(0,6) = -g1 + b8 * n55 - h55 * a8 * n55 ; TRANSX(8,2) = -l34 ;
TRANSX(0,7) = b8 * m55 - h55 * a8 * m55 + a8 * n55 - e2 ; TRANSX(8,3) = -k34 ;
TRANSX(0,8) = a8 * m55 - f8 - h55 * a8 * l55 + b8 * l55 ; TRANSX(8,4) = -i34 ;
TRANSX(0,9) = -h55 * a8 * k55 - e8 + b8 * k55 + a8 * l55 ; TRANSX(8,5) = -h34 ;
TRANSX(0,10) = -h55 * a8 * i55 - d8 + a8 * k55 + b8 * i55 ; TRANSX(9,0) = -n33 ;
TRANSX(0,11) = -c8 + b8 * h55 - h55 * h55 * a8 + a8 * i55 ; TRANSX(9,1) = -m33 ;
TRANSX(1,6) = b7 * n55 - h55 * a7 * n55 ; TRANSX(9,2) = -l33 ;
TRANSX(1,7) = b7 * m55 - h55 * a7 * m55 + a7 * n55 - e1 ; TRANSX(9,3) = -k33 ;
TRANSX(1,8) = a7 * m55 - f7 - h55 * a7 * l55 + b7 * l55 ; TRANSX(9,4) = -i33 ;
TRANSX(1,9) = -h55 * a7 * k55 + a7 * l55 - e7 + b7 * k55 ; TRANSX(9,5) = -h33 ;
TRANSX(1,10) = -h55 * a7 * i55 + a7 * k55 - d7 + b7 * i55 ; TRANSX(10,0) = -n32 ;
TRANSX(1,11) = -c7 + b7 * h55 - h55 * h55 * a7 + a7 * i55 ; TRANSX(10,1) = -m32 ;
TRANSX(2,6) = b6 * n55 - h55 * a6 * n55 ; TRANSX(10,2) = -l32 ;
TRANSX(2,7) = -h55 * a6 * m55 + a6 * n55 + b6 * m55 ; TRANSX(10,3) = -k32 ;
TRANSX(2,8) = -f6 - h55 * a6 * l55 + a6 * m55 + b6 * l55 ; TRANSX(10,4) = -i32 ;
TRANSX(2,9) = -h55 * a6 * k55 + b6 * k55 - e6 + a6 * l55 ; TRANSX(10,5) = -h32 ;
TRANSX(2,10) = -d6 + a6 * k55 - h55 * a6 * i55 + b6 * i55 ; TRANSX(11,0) = -n31 ;
TRANSX(2,11) = -c6 + b6 * h55 - h55 * h55 * a6 + a6 * i55 ; TRANSX(11,1) = -m31 ;
TRANSX(3,6) = b5 * n55 - h55 * a5 * n55 ; TRANSX(11,2) = -l31 ;
TRANSX(3,7) = -h55 * a5 * m55 + a5 * n55 + b5 * m55 ; TRANSX(11,3) = -k31 ;
TRANSX(3,8) = -f5 - h55 * a5 * l55 + a5 * m55 + b5 * l55 ; TRANSX(11,4) = -i31 ;
TRANSX(3,9) = -h55 * a5 * k55 + b5 * k55 - e5 + a5 * l55 ; TRANSX(11,5) = -h31 ;
TRANSX(3,10) = -d5 + a5 * k55 - h55 * a5 * i55 + b5 * i55 ; TRANSX(6,0) = -n36 ;
TRANSX(3,11) = -c5 + b5 * h55 - h55 * h55 * a5 + a5 * i55 ; TRANSX(6,1) = -m36 ;
TRANSX(4,6) = b4 * n55 - h55 * a4 * n55 ; TRANSX(6,2) = -l36 ;
TRANSX(4,7) = -h55 * a4 * m55 + a4 * n55 + b4 * m55 ; TRANSX(6,3) = -k36 ;
TRANSX(4,8) = -f4 - h55 * a4 * l55 + a4 * m55 + b4 * l55 ; TRANSX(6,4) = -i36 ;
TRANSX(4,9) = -h55 * a4 * k55 + b4 * k55 - e4 + a4 * l55 ; TRANSX(6,5) = -h36 ;
TRANSX(4,10) = -d4 + a4 * k55 - h55 * a4 * i55 + b4 * i55 ; TRANSX(7,0) = -n35 ;
TRANSX(4,11) = -c4 + a4 * i55 + b4 * h55 - h55 * h55 * a4 ; TRANSX(7,1) = -m35 ;
TRANSX(5,6) = b3 * n55 - h55 * a3 * n55 ; TRANSX(7,2) = -l35 ;
TRANSX(5,7) = -h55 * a3 * m55 + a3 * n55 + b3 * m55 ; TRANSX(7,3) = -k35 ;
TRANSX(5,8) = -f3 - h55 * a3 * l55 + a3 * m55 + b3 * l55 ; TRANSX(7,4) = -i35 ;
TRANSX(5,9) = -h55 * a3 * k55 + b3 * k55 - e3 + a3 * l55 ; TRANSX(7,5) = -h35 ;
TRANSX(5,10) = -d3 + a3 * k55 - h55 * a3 * i55 + b3 * i55 ; TRANSX(8,0) = -n34 ;
TRANSX(5,11) = -c3 + a3 * i55 + b3 * h55 - h55 * h55 * a3 ; TRANSX(8,1) = -m34 ;

calculValeursPropres(TRANSX,sol-Tx)

Matrice TRANSY(12,12)
TRANSY=0
// TY

TRANSY(0,6) = -g2 + b16 * n55 - h55 * a16 * n55 ; TRANSY(8,2) = -l43 ;
TRANSY(0,7) = b16 * m55 - h55 * a16 * m55 + a16 * n55 - e4 ; TRANSY(8,3) = -k43 ;
TRANSY(0,8) = a16 * m55 - f16 - h55 * a16 * l55 + b16 * l55 ; TRANSY(8,4) = -i43 ;

TRANSY(0,9) = -h55 * a16 * k55 - e16 + b16 * k55 + a16 * l55 ; TRANSY(8,5) = -h43 ;
TRANSY(0,10) = -h55 * a16 * i55 - d16 + a16 * k55 + b16 * i55 ; TRANSY(9,0) = -n42 ;
TRANSY(0,11) = -c16 + b16 * h55 - h55 * h55 * a16 + a16 * i55 ; TRANSY(9,1) = -m42 ;
TRANSY(1,6) = b15 * n55 - h55 * a15 * n55 ; TRANSY(9,2) = -l42 ;
TRANSY(1,7) = b15 * m55 - h55 * a15 * m55 + a15 * n55 - e3 ; TRANSY(9,3) = -k42 ;
TRANSY(1,8) = a15 * m55 - f15 - h55 * a15 * l55 + b15 * l55 ; TRANSY(9,4) = -i42 ;
TRANSY(1,9) = -h55 * a15 * k55 + a15 * l55 - e15 + b15 * k55 ; TRANSY(9,5) = -h42 ;
TRANSY(1,10) = -h55 * a15 * i55 + a15 * k55 - d15 + b15 * i55 ; TRANSY(10,0) = -n41 ;
TRANSY(1,11) = a15 * i55 + b15 * h55 - h55 * h55 * a15 - c15 ; TRANSY(10,1) = -m41 ;
TRANSY(2,6) = b14 * n55 - h55 * a14 * n55 ; TRANSY(10,2) = -l41 ;
TRANSY(2,7) = -h55 * a14 * m55 + a14 * n55 + b14 * m55 ; TRANSY(10,3) = -k41 ;
TRANSY(2,8) = -f14 - h55 * a14 * l55 + a14 * m55 + b14 * l55 ; TRANSY(10,4) = -i41 ;
TRANSY(2,9) = -h55 * a14 * k55 + b14 * k55 - e14 + a14 * l55 ; TRANSY(10,5) = -h41 ;
TRANSY(2,10) = -d14 + a14 * k55 - h55 * a14 * i55 + b14 * i55 ; TRANSY(11,0) = -n40 ;
TRANSY(2,11) = -c14 + b14 * h55 - h55 * h55 * a14 + a14 * i55 ; TRANSY(11,1) = -m40 ;
TRANSY(3,6) = b13 * n55 - h55 * a13 * n55 ; TRANSY(11,2) = -l40 ;
TRANSY(3,7) = -h55 * a13 * m55 + a13 * n55 + b13 * m55 ; TRANSY(11,3) = -k40 ;
TRANSY(3,8) = -f13 - h55 * a13 * l55 + a13 * m55 + b13 * l55 ; TRANSY(11,4) = -i40 ;
TRANSY(3,9) = -h55 * a13 * k55 + b13 * k55 - e13 + a13 * l55 ; TRANSY(11,5) = -h40 ;
TRANSY(3,10) = -d13 + a13 * k55 - h55 * a13 * i55 + b13 * i55 ; TRANSY(6,0) = -n45 ;
TRANSY(3,11) = -c13 + a13 * i55 + b13 * h55 - h55 * h55 * a13 ; TRANSY(6,1) = -m45 ;
TRANSY(4,6) = b12 * n55 - h55 * a12 * n55 ; TRANSY(6,2) = -l45 ;
TRANSY(4,7) = -h55 * a12 * m55 + a12 * n55 + b12 * m55 ; TRANSY(6,3) = -k45 ;
TRANSY(4,8) = -f12 - h55 * a12 * l55 + a12 * m55 + b12 * l55 ; TRANSY(6,4) = -i45 ;
TRANSY(4,9) = -h55 * a12 * k55 + b12 * k55 - e12 + a12 * l55 ; TRANSY(6,5) = -h45 ;
TRANSY(4,10) = -d12 + a12 * k55 - h55 * a12 * i55 + b12 * i55 ; TRANSY(7,0) = -n44 ;
TRANSY(4,11) = -c12 + b12 * h55 - h55 * h55 * a12 + a12 * i55 ; TRANSY(7,1) = -m44 ;
TRANSY(5,6) = b11 * n55 - h55 * a11 * n55 ; TRANSY(7,2) = -l44 ;
TRANSY(5,7) = -h55 * a11 * m55 + a11 * n55 + b11 * m55 ; TRANSY(7,3) = -k44 ;
TRANSY(5,8) = -f11 - h55 * a11 * l55 + a11 * m55 + b11 * l55 ; TRANSY(7,4) = -i44 ;
TRANSY(5,9) = -h55 * a11 * k55 + b11 * k55 - e11 + a11 * l55 ; TRANSY(7,5) = -h44 ;
TRANSY(5,10) = -d11 + a11 * k55 - h55 * a11 * i55 + b11 * i55 ; TRANSY(8,0) = -n43 ;
TRANSY(5,11) = -c11 + b11 * h55 - h55 * h55 * a11 + a11 * i55 ; TRANSY(8,1) = -m43 ;

calculValeursPropres(TRANSY,sol-Ty)

Matrice TRANSZ(12,12)
TRANSZ=0
//TZ

TRANSZ(0,6) = 1 ; TRANSZ(8,1) = -m52 ;
TRANSZ(1,7) = 1 ; TRANSZ(8,2) = -l52 ;
TRANSZ(2,8) = 1 ; TRANSZ(8,3) = -k52 ;
TRANSZ(3,9) = 1 ; TRANSZ(8,4) = -i52 ;
TRANSZ(4,10) = 1 ; TRANSZ(8,5) = -h52 ;
TRANSZ(5,11) = 1 ; TRANSZ(9,0) = -n51 ;
TRANSZ(6,0) = -n54 ; TRANSZ(9,1) = -m51 ;
TRANSZ(6,1) = -m54 ; TRANSZ(9,2) = -l51 ;
TRANSZ(6,2) = -l54 ; TRANSZ(9,3) = -k51 ;
TRANSZ(6,3) = -k54 ; TRANSZ(9,4) = -i51 ;
TRANSZ(6,4) = -i54 ; TRANSZ(9,5) = -h51 ;
TRANSZ(6,5) = -h54 ; TRANSZ(10,0) = -n50 ;
TRANSZ(7,0) = -n53 ; TRANSZ(10,1) = -m50 ;
TRANSZ(7,1) = -m53 ; TRANSZ(10,2) = -l50 ;
TRANSZ(7,2) = -l53 ; TRANSZ(10,3) = -k50 ;
TRANSZ(7,3) = -k53 ; TRANSZ(10,4) = -i50 ;
TRANSZ(7,4) = -i53 ; TRANSZ(10,5) = -h50 ;
TRANSZ(7,5) = -h53 ; TRANSZ(11,0) = -n49 ;
TRANSZ(8,0) = -n52 ; TRANSZ(11,1) = -m49 ;

TRANSZ(11,4) = -i49 ; TRANSZ(11,2) = -l49 ;
TRANSZ(11,5) = -h49 ; TRANSZ(11,3) = -k49 ;

calculValeursPropres(TRANSZ,sol-Tz)

## C.1 Conclusion

La matrice M décrite dans cette annexe est calculée une fois pour toutes. A chaque nouveau jeu de coordonnées, on sait exactement où elles doivent se positionner dans cette matrice. Ensuite nous avons vu les matrices action pour chaque inconnue. Le calcul des valeurs propres de ces matrices d'action donne directement les solutions des inconnues.

# D

# La méthode des 7 paramètres : calcul direct de la similitude 3D

Dans cette annexe nous décrivons la matrice de Macaulay, que nous avons fabriquée afin de résoudre les équations décrites dans le chapitre 8. Cette matrice que nous appelons M, est une matrice qui a 24 lignes et 28 colonnes. En effectuant une élimination du type Gauss-Jordan sur cette matrice, nous obtenons les bases de Gröbner. Dans la suite, nous donnons les matrices d'action (Mlambda, Ma, Mb, Mc) pour chacune des inconnues. Le calcul des valeurs propres de cette matrice donne la solution pour chaque inconnue.

Matrice M(24,28)

M=0

M(0,17) = Yb1 - Yc1 ; M(8,16) = Ya2 - Yc2 ; M(4,3) = Yb1 - Yc1 ; M(12,23) = -Za2 + Zb2 ;
M(0,19) = -Xb1 + Xc1 ; M(8,18) = -Xa2 + Xc2 ; M(4,5) = -Xb1 + Xc1 ; M(12,25) = Xa2 - Xb2 ;
M(0,23) = Yb2 - Yc2 ; M(8,21) = Za1 - Zc1 ; M(4,9) = Yb2 - Yc2 ; M(12,26) = Ya1 - Yb1 ;
M(0,24) = -Xb2 + Xc2 ; M(8,25) = -Za2 + Zc2 ; M(4,11) = -Xb2 + Xc2 ; M(12,27) = -Ya2 + Yb2 ;
M(0,26) = Zb1 - Zc1 ; M(9,4) = Ya1 - Yc1 ; M(4,14) = Zb1 - Zc1 ; M(13,10) = -Za1 + Zb1 ;
M(0,27) = -Zb2 + Zc2 ; M(9,6) = -Xa1 + Xc1 ; M(4,21) = -Zb2 + Zc2 ; M(13,14) = Xa1 - Xb1 ;
M(1,10) = Yb1 - Yc1 ; M(9,10) = Ya2 - Yc2 ; M(5,0) = Yb1 - Yc1 ; M(13,17) = -Za2 + Zb2 ;
M(1,12) = -Xb1 + Xc1 ; M(9,12) = -Xa2 + Xc2 ; M(5,1) = -Xb1 + Xc1 ; M(13,21) = Xa2 - Xb2 ;
M(1,17) = Yb2 - Yc2 ; M(9,15) = Za1 - Zc1 ; M(5,3) = Yb2 - Yc2 ; M(13,22) = Ya1 - Yb1 ;
M(1,19) = -Xb2 + Xc2 ; M(9,22) = -Za2 + Zc2 ; M(5,5) = -Xb2 + Xc2 ; M(13,26) = -Ya2 + Yb2 ;
M(1,22) = Zb1 - Zc1 ; M(10,3) = Ya1 - Yc1 ; M(5,8) = Zb1 - Zc1 ; M(14,9) = -Za1 + Zb1 ;
M(1,26) = -Zb2 + Zc2 ; M(10,5) = -Xa1 + Xc1 ; M(5,14) = -Zb2 + Zc2 ; M(14,13) = Xa1 - Xb1 ;
M(2,9) = Yb1 - Yc1 ; M(10,9) = Ya2 - Yc2 ; M(6,17) = Ya1 - Yc1 ; M(14,16) = -Za2 + Zb2 ;
M(2,11) = -Xb1 + Xc1 ; M(10,11) = -Xa2 + Xc2 ; M(6,19) = -Xa1 + Xc1 ; M(14,20) = Xa2 - Xb2 ;
M(2,16) = Yb2 - Yc2 ; M(10,14) = Za1 - Zc1 ; M(6,23) = Ya2 - Yc2 ; M(14,21) = Ya1 - Yb1 ;
M(2,18) = -Xb2 + Xc2 ; M(10,21) = -Za2 + Zc2 ; M(6,24) = -Xa2 + Xc2 ; M(14,25) = -Ya2 + Yb2 ;
M(2,21) = Zb1 - Zc1 ; M(11,0) = Ya1 - Yc1 ; M(6,26) = Za1 - Zc1 ; M(15,4) = -Za1 + Zb1 ;
M(2,25) = -Zb2 + Zc2 ; M(11,1) = -Xa1 + Xc1 ; M(6,27) = -Za2 + Zc2 ; M(15,8) = Xa1 - Xb1 ;
M(3,4) = Yb1 - Yc1 ; M(11,3) = Ya2 - Yc2 ; M(7,10) = Ya1 - Yc1 ; M(15,10) = -Za2 + Zb2 ;
M(3,6) = -Xb1 + Xc1 ; M(11,5) = -Xa2 + Xc2 ; M(7,12) = -Xa1 + Xc1 ; M(15,14) = Xa2 - Xb2 ;
M(3,10) = Yb2 - Yc2 ; M(11,8) = Za1 - Zc1 ; M(7,17) = Ya2 - Yc2 ; M(15,15) = Ya1 - Yb1 ;
M(3,12) = -Xb2 + Xc2 ; M(11,14) = -Za2 + Zc2 ; M(7,19) = -Xa2 + Xc2 ; M(15,22) = -Ya2 + Yb2 ;
M(3,15) = Zb1 - Zc1 ; M(12,17) = -Za1 + Zb1 ; M(7,22) = Za1 - Zc1 ; M(16,3) = -Za1 + Zb1 ;
M(3,22) = -Zb2 + Zc2 ; M(12,21) = Xa1 - Xb1 ; M(7,26) = -Za2 + Zc2 ; M(16,7) = Xa1 - Xb1 ;
M(8,11) = -Xa1 + Xc1 ; M(16,13) = Xa2 - Xb2 ; M(8,9) = Ya1 - Yc1 ; M(16,9) = -Za2 + Zb2 ;

M= ElminationGauss(M)

a1=M(0,18) ; b8=M(11,20) ; d4=M(3,27) ; d16=M(16,27) ;
b1=M(0,20) ; c12=M(11,26) ; c5=M(4,26) ; c18=M(17,26) ;
c1=M(0,26) ; d12=M(11,27) ; d5=M(4,27) ; d17=M(17,27) ;

```
d1=M(0,27) ;  c13=M(12,26) ; a5=M(5,18) ; c19=M(18,26) ;
a2=M(1,18) ;  d13=M(12,27) ; b5=M(5,20) ; d18=M(18,27) ;
b2=M(1,20) ;   a9=M(13,18) ; c6=M(5,26) ; c20=M(19,26) ;
c2=M(1,26) ;   b9=M(13,20) ; d6=M(5,27) ; d19=M(19,27) ;
d2=M(1,27) ;  c14=M(13,26) ; c7=M(6,26) ; d20=M(20,27) ;
a3=M(2,18) ;  d14=M(13,27) ; d7=M(6,27) ; c21=M(21,26) ;
b3=M(2,20) ;  c15=M(14,26) ; a6=M(7,18) ; d21=M(21,27) ;
c3=M(2,26) ;  d15=M(14,27) ; b6=M(7,20) ; c22=M(22,26) ;
d3=M(2,27) ;  c16=M(15,26) ; c8=M(7,26) ; d22=M(22,27) ;
a4=M(3,18) ;  a10=M(16,18) ; d8=M(7,27) ; c23=M(23,26) ;
b4=M(3,20) ;  b10=M(16,20) ; c9=M(8,26) ; d23=M(23,27) ;
c4=M(3,26) ;  c17=M(16,26) ; d9=M(8,27) ;  c10=M(9,26) ;
a7=M(9,18) ;   d10=M(9,27) ; b7=M(9,20) ; c11=M(10,26) ;
d11=M(10,27) ; a8=M(11,18) ;
```

```
    // calcul des matrices action de chaque inconnues
    // lambda Matrice Mlambda(2,2)
Mlambda=0

    Mlambda(0,1) = 1
Mlambda(1,0) = -d20

    calculValeursPropres(Mlambda,sol-lambda)

    // a Matrice Ma(2,2)
Ma=0
Ma(0,0) = -d21
Ma(0,1) = -c21
Ma(1,0) = -d17
Ma(1,1) = -c18

    calculValeursPropres(Ma,sol-a)

    //b Matrice Mb(2,2)
Mb=0
Mb(0,0) = -d22
Mb(0,1) = -c22
Mb(1,0) = -d18
Mb(1,1) = -c19

    calculValeursPropres(Mb,sol-b)

    // c Matrice Mc(2,2)
Mc=0
Mc(0,0) = -d23
Mc(0,1) = -c23
Mc(1,0) = -d19
Mc(1,1) = -c20

    calculValeursPropres(Mc,sol-c)
```

## D.1 Conclusion

La matrice M décrite dans cette annexe est calculée une fois pour toutes. A chaque nouveau jeu de coordonnées, on sait exactement où elles doivent se positionner dans cette matrice. Ensuite nous avons vu les matrices action pour chaque inconnue. Le calcul des valeurs propres de ces matrices d'action donne directement les solutions des inconnues.

# E

# Résulat des Simulations de la méthode des 7 paramètres

Dans cette annexe nous présentons les résultats des simulations, afin de valider la méthode des 7 paramètres mis en place dans le chapitre 9.

### E.0.1 Variation de la rotation

Dans cette configuration l'angle de rotation sur chacun des axes varie à son tour. La translation est égale à (2000, 1000, 500). Le facteur d'échelle est égal à 1.5. Nous avons fait varier l'angle de rotation sur l'axe des X, Y, et Z, allant de 0 à 80 degrés, avec un pas de 20 degrés. Dans chaque cas une seule des composantes de la rotation varie, et les deux autres restent à zéro.

*Variation de l'angle de rotation sur l'axe des X*

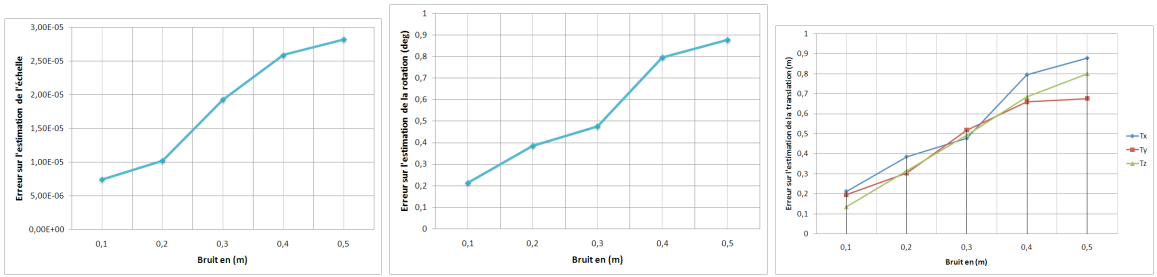



**Fig. E.1.** Rotation (0° 0° 0°)

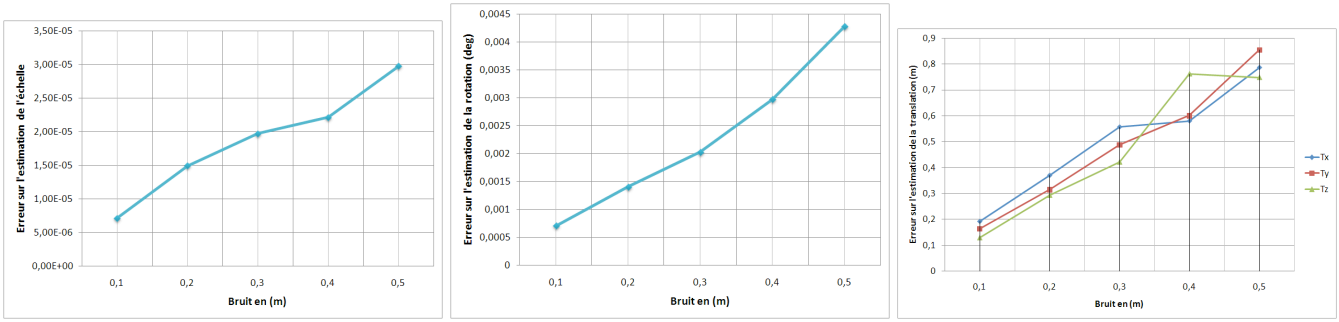



**Fig. E.2.** Rotation (20˚ 0˚ 0˚)

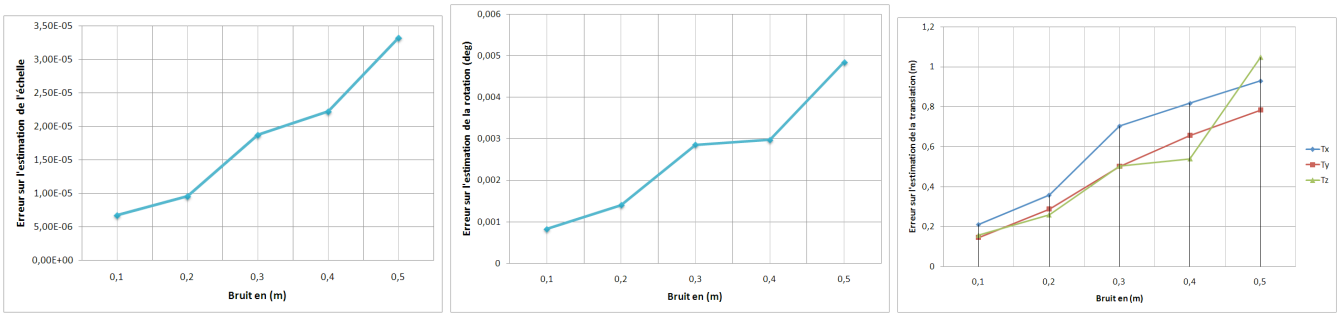



**Fig. E.3.** Rotation (40˚ 0˚ 0˚)

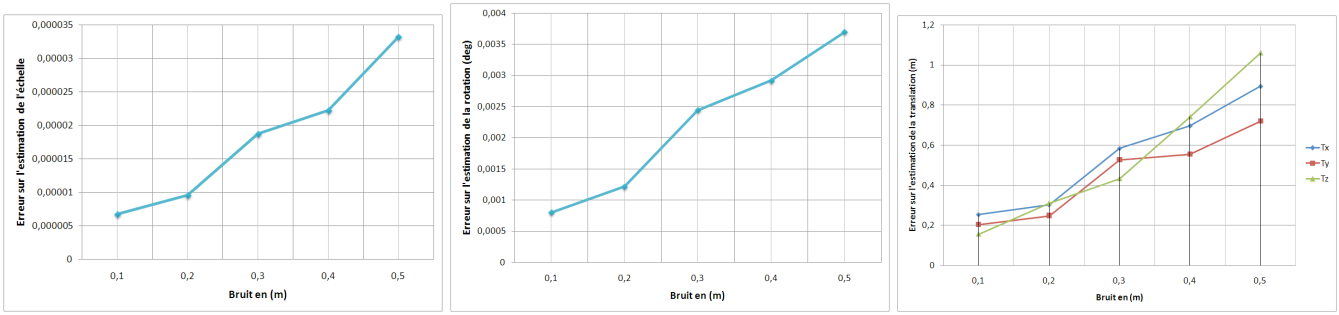



**Fig. E.4.** Rotation (60˚ 0˚ 0˚)

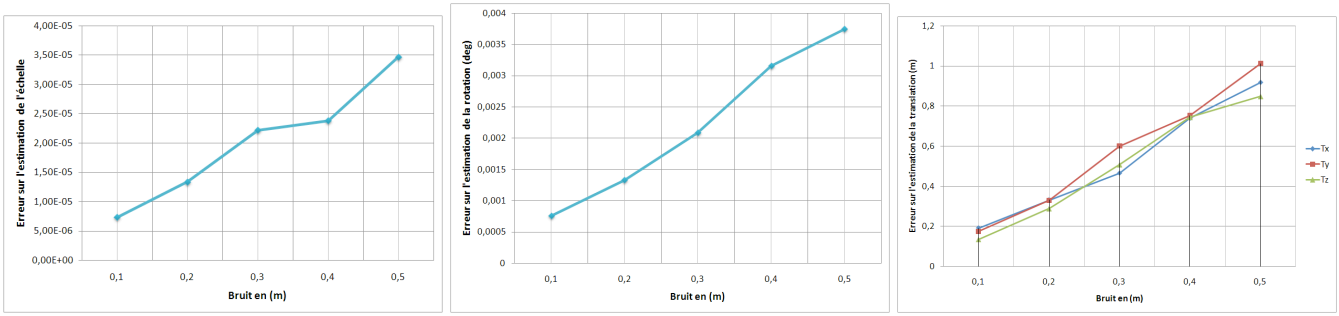



**Fig. E.5.** Rotation (80° 0° 0°)

*Variation de l'angle de rotation sur l'axe des Y*

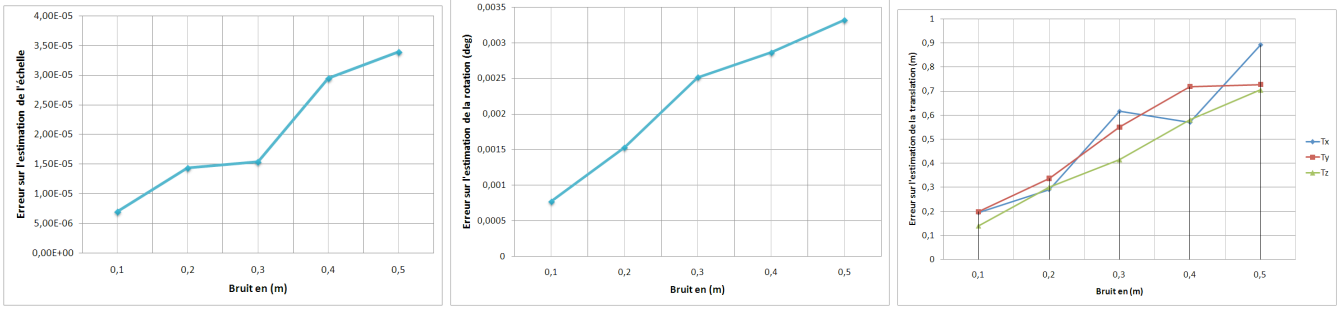



**Fig. E.6.** Rotation (0° 20° 0°)

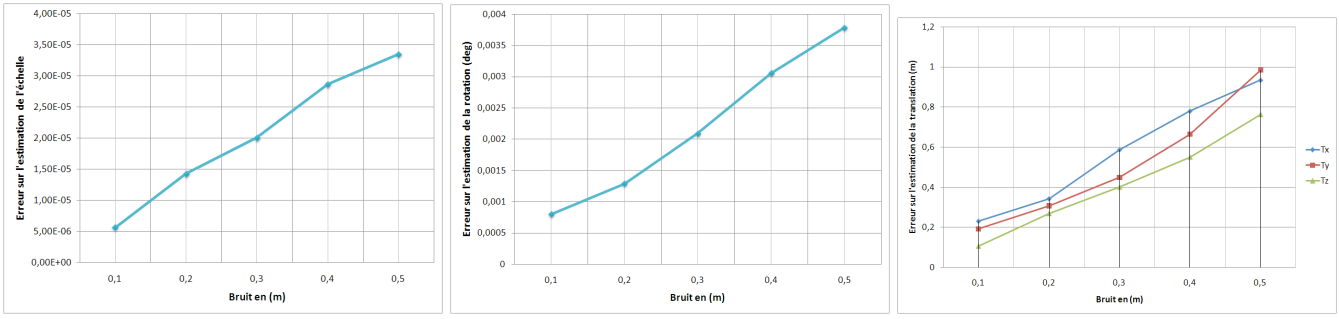



**Fig. E.7.** Rotation (0° 40° 0°)

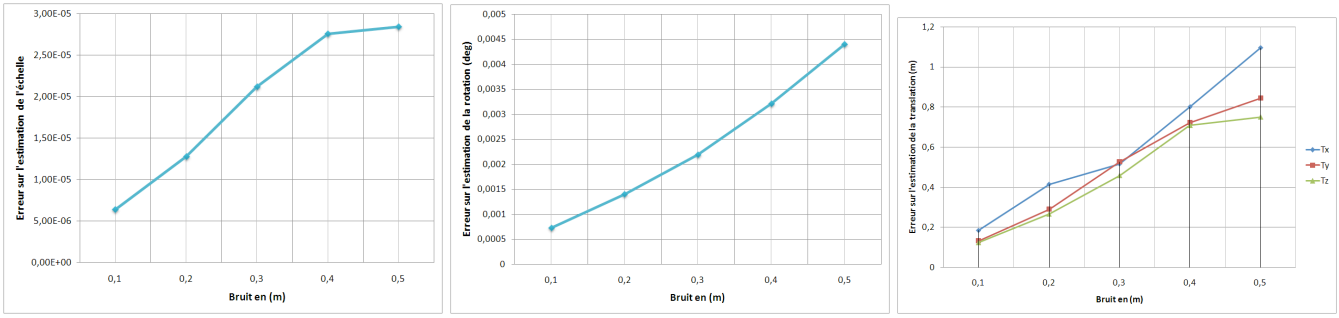



**Fig. E.8.** Rotation (0° 60° 0°)

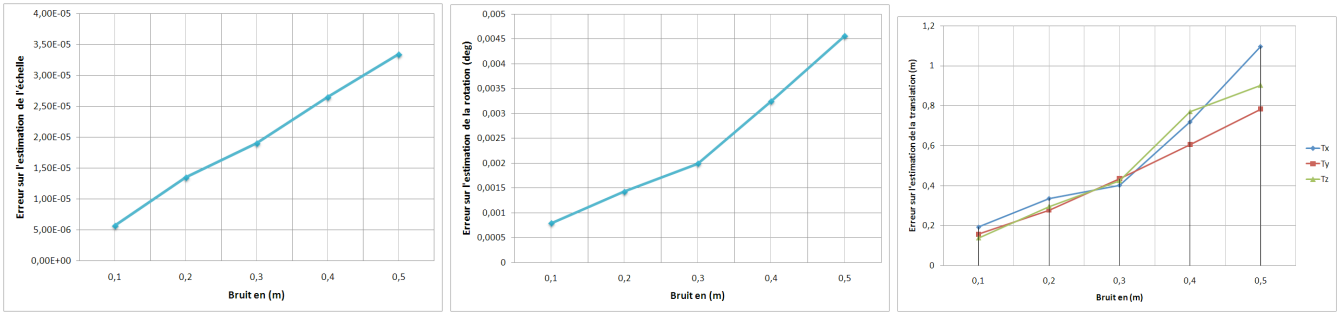



**Fig. E.9.** Rotation (0° 80° 0°)

*Variation de l'angle de rotation sur l'axe des Z*

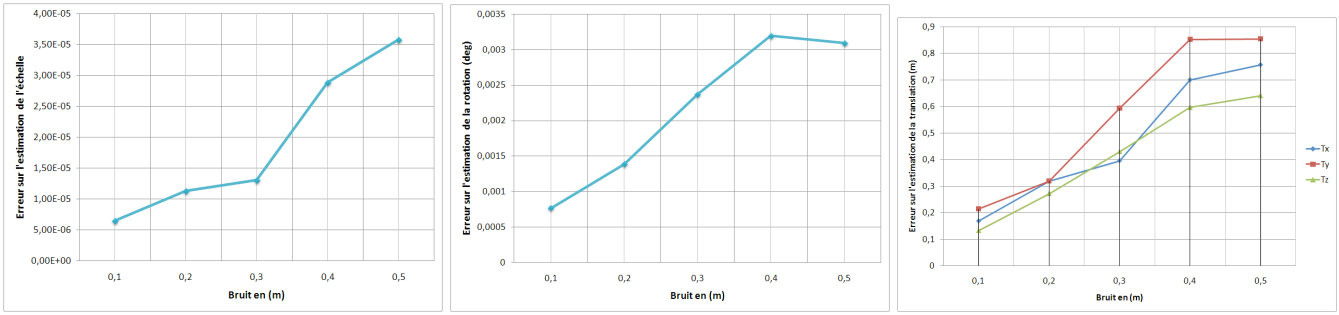



**Fig. E.10.** Rotation (0° 0° 20°)

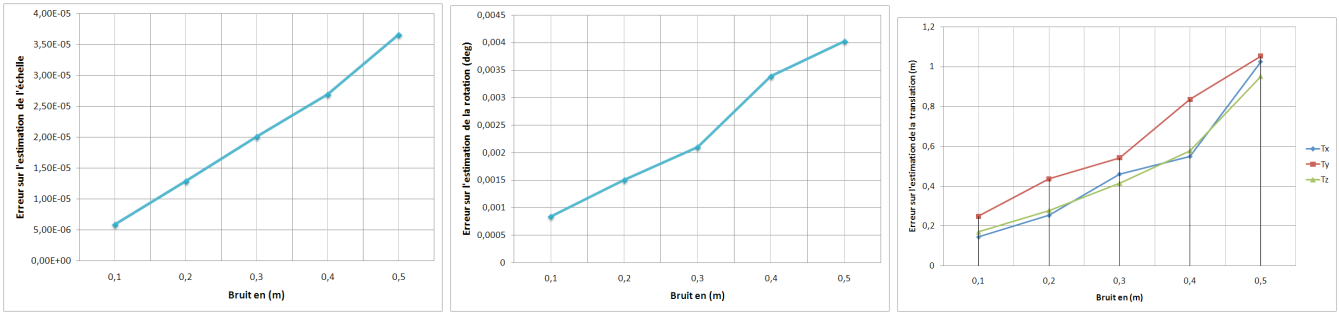



**Fig. E.11.** Rotation (0° 0° 40°)

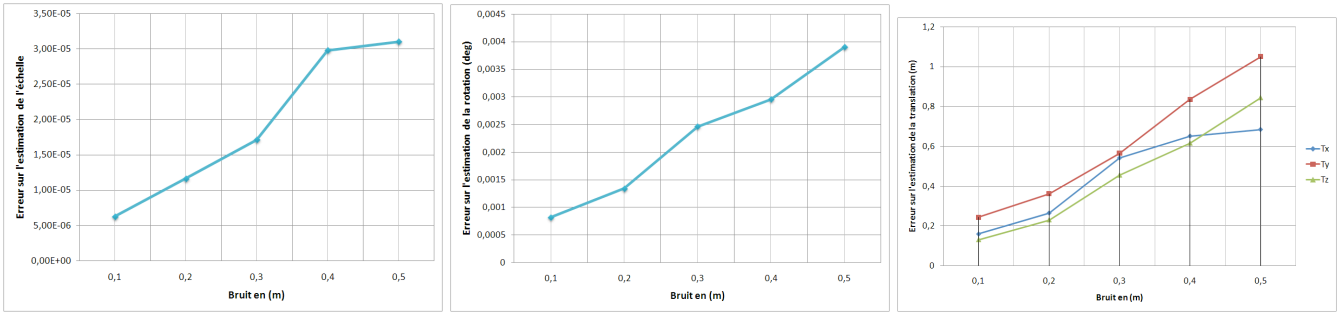



**Fig. E.12.** Rotation (0° 0° 60°)

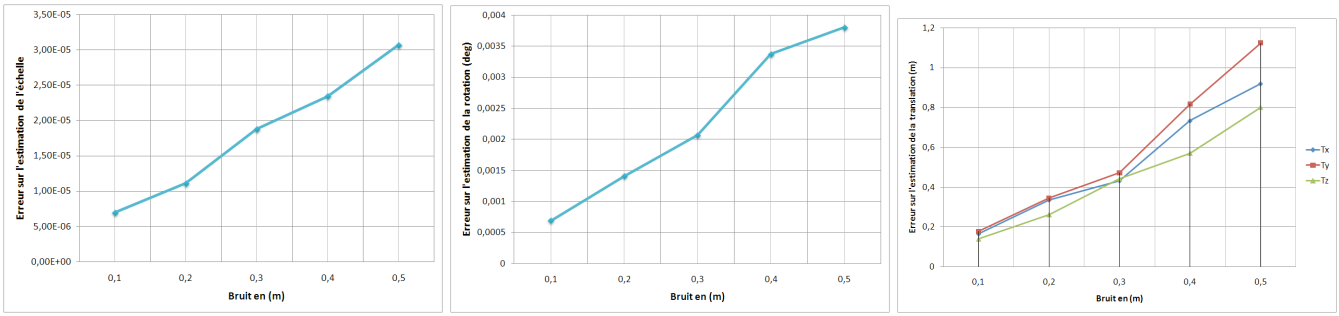



**Fig. E.13.** Rotation (0° 0° 80°)

## E.0.2 Variation de la translation

Dans cette configuration les trois angles de rotation ne varient pas et sont égaux à 40°, 20° et 30°. Le facteur d'échelle est égal à 2. La translation quant à elle varie sur chaque axe de 0 à 6000, avec un pas de 2000. Le cas de translation 0 est celui d'une configuration panoramique.

## *Variation de la translation en X*


**Fig. E.14.** Translation (0 0 0)

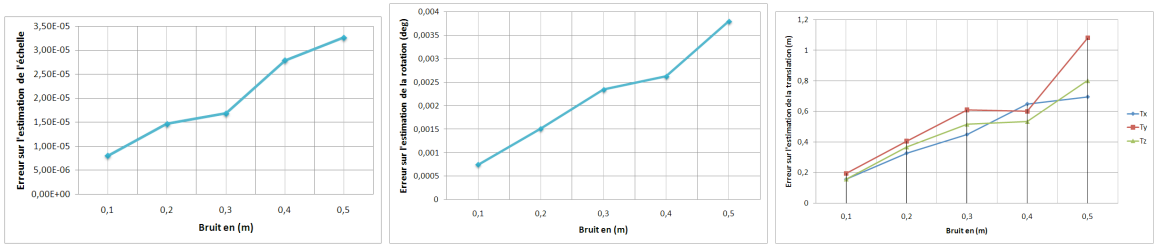



**Fig. E.15.** Translation (2000 0 0)

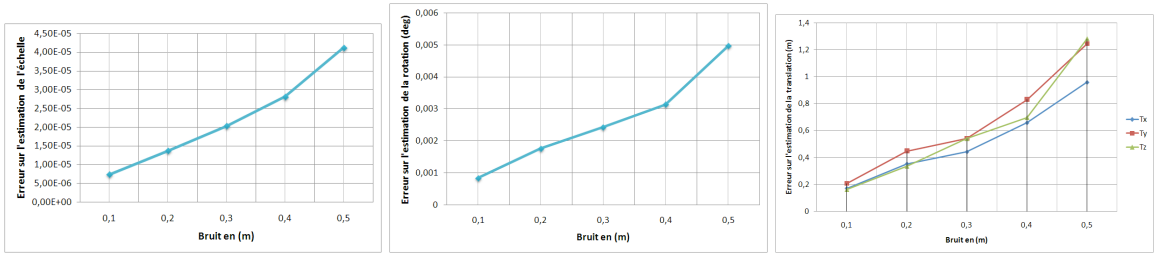



**Fig. E.16.** Translation (4000 0 0)

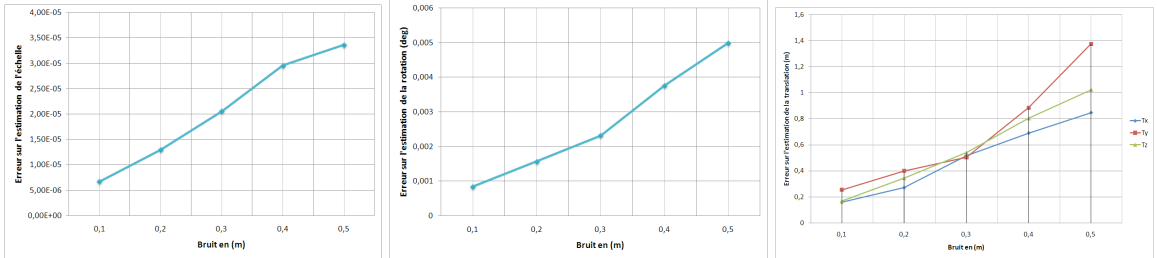



**Fig. E.17.** Translation (6000 0 0)

*Variation de la translation en Y*

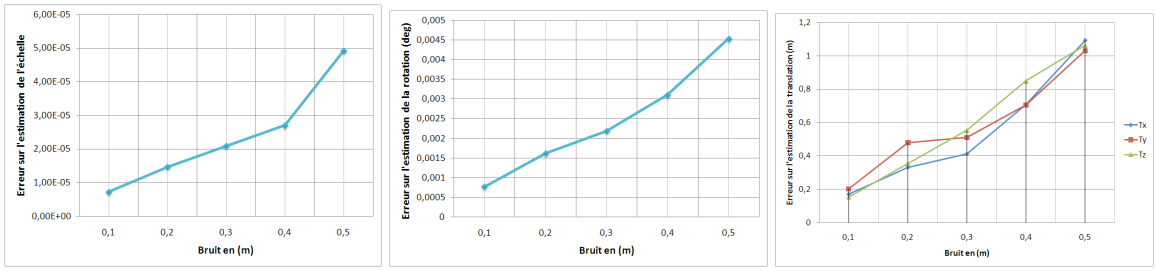



**Fig. E.18.** Translation (0 2000 0)

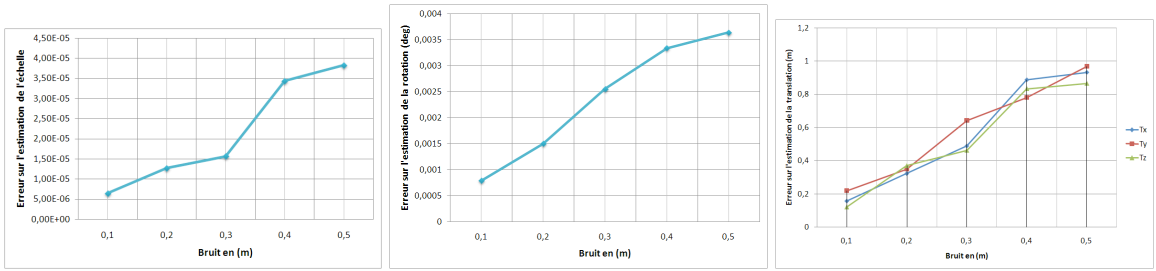



**Fig. E.19.** Translation (0 4000 0)

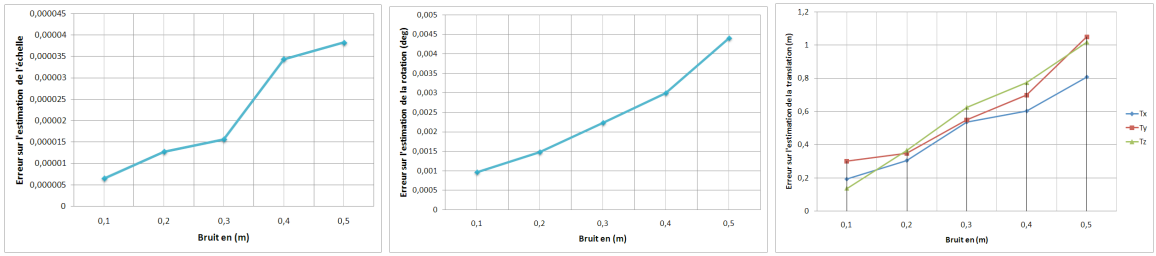



**Fig. E.20.** Translation (0 6000 0)

*Variation de la translation en Z*

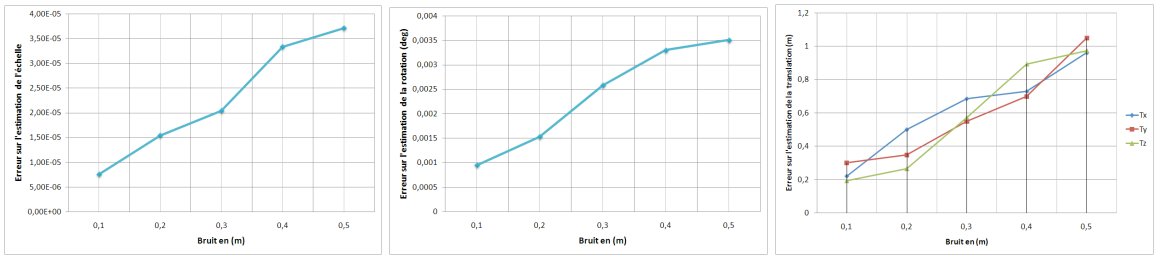



**Fig. E.21.** Translation (0 0 2000)

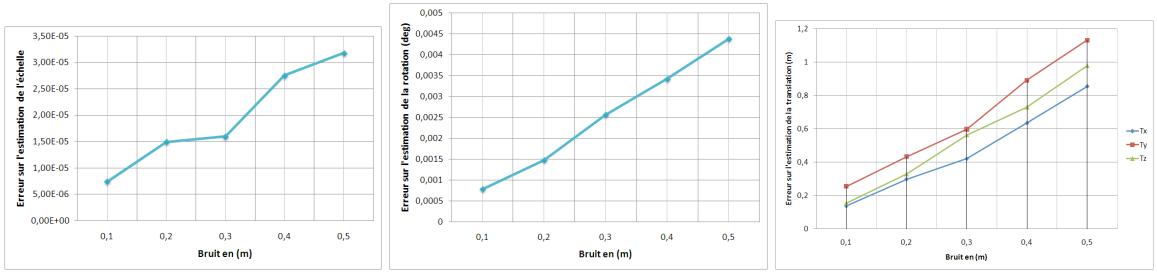



**Fig. E.22.** Translation (0 0 4000 )

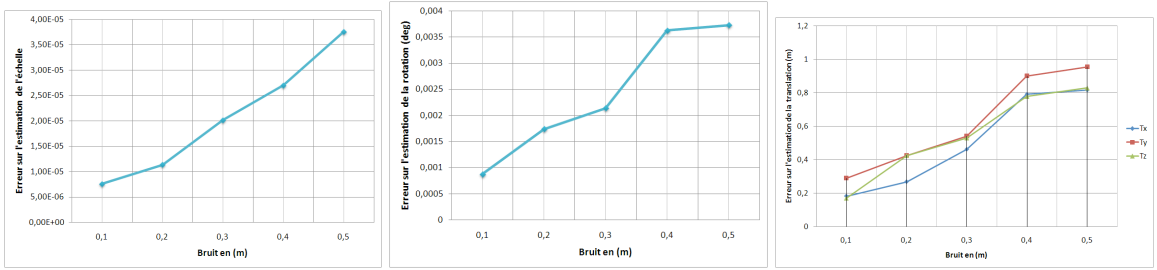



**Fig. E.23.** Translation (0 0 6000)

### E.0.3 Variation de l'échelle

Dans cette configuration seule l'échelle varie. 4 valeurs sont prises en compte : 0.5, 1, 1.5 et 2. Les valeurs de la rotation sont égales à 40 degrés, 20 degrés et 30 degrés. La translation est égale à (2000, 1000, 500).

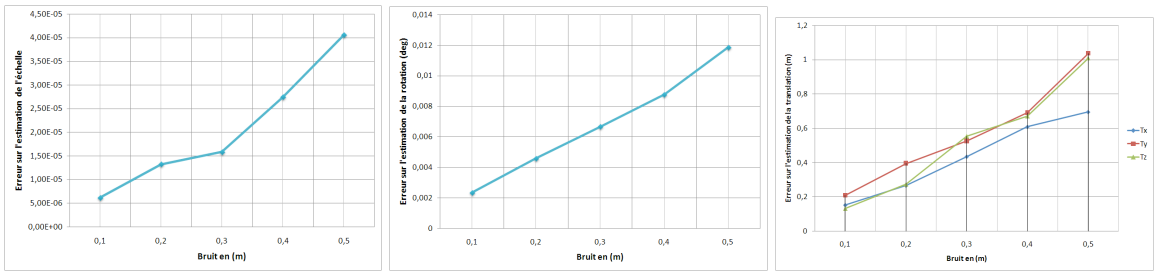



**Fig. E.24.** échelle = 0.5

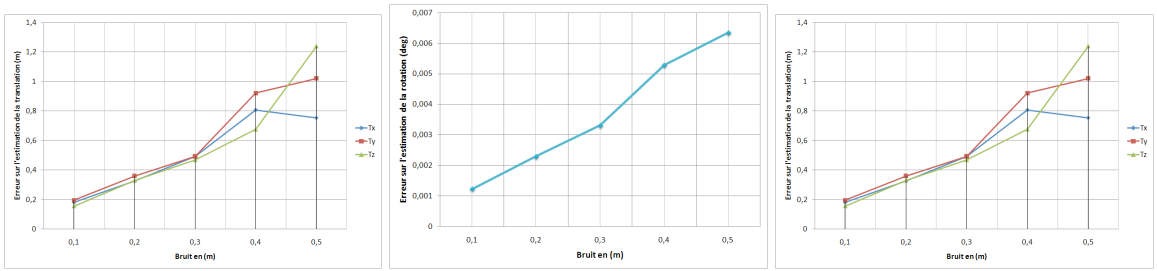



**Fig. E.25.** échelle = 1

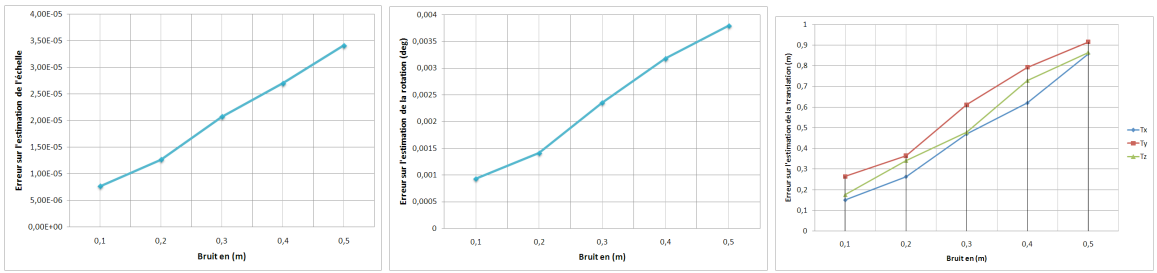



**Fig. E.26.** échelle = 1.5

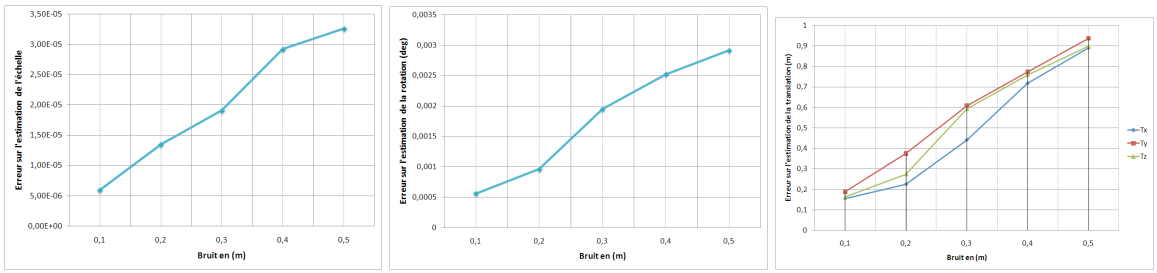



**Fig. E.27.** échelle = 2